



PROJECT NO. 72

A RESTAURANT RECOMMENDATION SYSTEM FOR INDIVIDUALS AND GROUPS

MR. CHANCHANA WICHA

MS. IRIN YOOKTAJARONG

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR

THE DEGREE OF BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)

FACULTY OF ENGINEERING

KING MONGKUT'S UNIVERSITY OF TECHNOLOGY THONBURI

2020

PROJECT NO. 72

A RESTAURANT RECOMMENDATION SYSTEM FOR INDIVIDUALS AND GROUPS

MR. CHANCHANA WICHA

MS. IRIN YOOKTAJARONG

A Project Submitted in Partial Fulfillment

of the Requirements for

the Degree of Bachelor of Engineering (Computer Engineering)

Faculty of Engineering

King Mongkut's University of Technology Thonburi

2020

Project Committee

.....
(Dr. Unchalisra Taetragool, Ph.D.)

Project Advisor

.....
(Asst.Prof.Dr. Rajchawit Sarochawikasit, Ph.D.)

Committee Member

.....
(Dr. Sally E. Goldin, Ph.D.)

Committee Member

.....
(Asst.Prof.Mr. Sanan Srakaew, B.Eng.)

Committee Member

Project Title	PROJECT NO. 72 A RESTAURANT RECOMMENDATION SYSTEM FOR INDIVIDUALS AND GROUPS
Credits	3
Member(s)	MR. CHANCHANA WICHA MS. IRIN YOOKTAJARONG
Project Advisor	Dr. Unchalisra Taetragool, Ph.D.
Program	Bachelor of Engineering
Field of Study	Computer Engineering
Department	Computer Engineering
Faculty	Engineering
Academic Year	2020

Abstract

In recent years, recommendation systems are widely used for movies, music, or products in order to reduce the time and effort of the users' searching and increase users' experience. However, restaurant recommendation systems are still very few in this industry and the existing recommenders in Thailand are just randomization applications and the majority of recommender systems are designed for individual users. In contrast, since this activity or event usually occurs in a group of people, not just individually, which each person can have the same or different behaviors, preferences, and interests. Therefore, group decisions often differ from individual decisions. As a result, each decision is often taken longer than one person, and sometimes conclusions are not able to be drawn or it may cause conflicts within the group.

According to the mentioned problems, we were motivated to develop this project, A Restaurant Recommendation System for Individuals and Groups, to better assist in group decision making. Our project is a web application that recommends restaurants in the Bangkok area to users based on their behaviors, preferences, and interests of both individuals and groups eating. The genetic algorithm is applied for generating suitable recommended restaurants for users. Therefore, this project aims to reduce time-wasting of decision making for both individuals and groups and increase the users' experience and satisfaction when using our recommendation system.

Keywords: Genetic Algorithm / Optimization Problem / Decision Making Process / Group Recommendation Systems / Web Application / Restaurants

หัวข้อปริญญาบัตร	หัวข้อปริญญาบัตรทั้งหมด
หน่วยกิต	3
ผู้เขียน	นายชาญชนา วิชา นางสาวไอริน ยุกตารงค์
อาจารย์ที่ปรึกษา	ดร. อัญชลิสา แต่ตระกูล
หลักสูตร	วิศวกรรมศาสตรบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมคอมพิวเตอร์
คณะ	วิศวกรรมศาสตร์
ปีการศึกษา	2563

บทคัดย่อ

ในช่วงไม่กี่ปีที่ผ่านมาได้มีการใช้ระบบแนะนำ (Recommendation System) อย่างแพร่หลายไม่ว่าจะเป็นภาคพยนตร์ เพลง หรือระบบแนะนำผลิตภัณฑ์ต่าง ๆ เพื่อลดระยะเวลาและความยุ่งยากในการค้นหาของผู้ใช้รวมถึงเพิ่มประสบการณ์ของผู้ใช้อย่างไร้ตัวมาระบบแนะนำร้านอาหาร ยังมีน้อยมากในวงการระบบคำแนะนำจากนั้นระบบแนะนำร้านอาหารที่มีอยู่ในประเทศไทยเป็นเพียงแอปพลิเคชันแบบสุ่มและส่วนใหญ่ออกแบบมาสำหรับผู้ใช้แต่ละรายเท่านั้น ซึ่งในความเป็นจริงแล้วการรับประทานอาหารมักเกิดขึ้นเป็นกลุ่มตั้งแต่ 2 คนขึ้นไปไม่ได้เป็นรายบุคคลเพียงอย่างเดียว ซึ่งแต่ละคนในกลุ่มนั้นมีพฤติกรรม ความชอบและความสนใจที่อาจจะเหมือนหรือแตกต่างกันดังนั้นการตัดสินใจของกลุ่มมักแตกต่างจากการตัดสินใจของแต่ละบุคคล ด้วยเหตุนี้การตัดสินใจในแต่ละครั้ง มักใช้เวลานานกว่าการตัดสินใจด้วยคน ๆ เดียว และในบางครั้งการตัดสินใจนั้นอาจจะไม่สามารถหาข้อมูลได้ หรืออาจทำให้เกิดความขัดแย้งขึ้นภายในกลุ่ม

ด้วยเหตุนี้เอง ทางผู้พัฒนาจึงมีความตั้งใจที่จะพัฒนาเว็บแอปพลิเคชันสำหรับแนะนำร้านอาหารที่อยู่ในพื้นที่จังหวัดกรุงเทพมหานครสำหรับรายบุคคลและรายกลุ่มขึ้น เพื่อเป็นตัวช่วยในการตัดสินใจทั้งแบบเดี่ยวและกลุ่มได้ดีมากยิ่งขึ้น โดยระบบแนะนำร้านอาหารจะอ้างอิงจากข้อมูลของผู้ใช้งาน ไม่ว่าจะเป็นพฤติกรรม ความสนใจ หรือความชอบ ทั้งแบบเดี่ยวและกลุ่ม ด้วยขั้นตอนวิธีเชิงพันธุกรรม (Genetic Algorithm) ในการหาค่าเหมาะสมที่สุดเชิงคณิตศาสตร์ (Mathematical Optimization) สุดท้ายนี้ผู้พัฒนามีความมุ่งหวังว่าระบบแนะนำร้านอาหารรายบุคคลและรายกลุ่มในโครงงานนี้จะช่วยลดเวลาที่เสียไปในการตัดสินใจและเพิ่มความพึงพอใจของผู้ใช้งานต่อร้านอาหารที่ระบบได้ทำการแนะนำ

คำสำคัญ: ขั้นตอนวิธีเชิงพันธุกรรม / การหาค่าเหมาะสมที่สุดเชิงคณิตศาสตร์ / กระบวนการตัดสินใจ / ระบบแนะนำรายกลุ่ม / เว็บแอปพลิเคชัน / ร้านอาหาร

ACKNOWLEDGMENTS

Throughout the writing of this project we have received a great deal of support and assistance.

We would like to express my very great appreciation to Dr. Unchalis Taetragool for her valuable and constructive suggestions during the planning and development of this project work. Her willingness to give his time so generously has been very much appreciated.

We would also like to express our very great appreciation to our committeees, Asst.Prof.Dr. Rajchawit Sarochawikasit, Dr. Sally E. Goldin and Asst.Prof.Mr. Sanan Srakaew, for their valuable guidelines and suggestions throughout our studies.

We also appreciate the Department of Computer Engineering for providing the laboratory and resources for the implementation of this project.

In addition, we would like to thank our parents for their wise counsel and sympathetic ear. You are always there for us. We could not have completed this project without the support of my friends who provided the feedback and suggestions throughout the development process. Finally, we would like to thank Avril Lavigne for entertaining us while we were writing this report.

CONTENTS

	PAGE
ABSTRACT	ii
THAI ABSTRACT	iii
ACKNOWLEDGMENTS	iv
CONTENTS	x
LIST OF TABLES	xi
LIST OF FIGURES	xii
LIST OF SYMBOLS	xiv
LIST OF TECHNICAL VOCABULARY AND ABBREVIATIONS	xv
 CHAPTER	
1. INTRODUCTION	1
1.1 Keywords	1
1.2 Problem Statement, Motivation, and Potential Benefits	1
1.3 Project Type	1
1.4 Proposed Method	1
1.4.1 Approach	1
1.4.1.1 Literature Review and Survey	2
1.4.1.2 Data Collection	2
1.4.1.3 Data Preparation	2
1.4.1.4 Construct the Model	2
1.4.1.5 Design User Interface	2
1.4.1.6 Build a Web Application	2
1.4.1.7 Validation	2
1.4.2 Objectives	2
1.4.3 Scope	3
1.5 Original Engineering Content	3
1.5.1 Web Application	3
1.5.2 Customer Behavior Dataset	3

1.5.3 Group Recommendation Algorithm	3
1.6 Task Breakdown and Draft Schedule	3
1.6.1 Semester 1	3
1.6.2 Semester 2	4
1.7 Deliverables for Term 1	7
1.8 Deliverables for Term 2	7
2. BACKGROUND THEORY AND RELATED WORK	9
2.1 Background	9
2.2 Related Theory	9
2.2.1 Web Application	9
2.2.2 Information Collection	9
2.2.2.1 Explicit Feedback	10
2.2.2.2 Implicit Feedback	10
2.2.3 Overview of Machine Learning	11
2.2.4 Recommendation System Techniques	11
2.2.4.1 Content-based Filtering Technique	11
2.2.4.2 Collaborative Filtering Technique	12
2.2.4.2.1 Memory-based	12
2.2.4.2.2 Model-based	13
2.2.4.3 Hybrid Filtering Technique	13
2.2.5 Aggregation Mechanism	14
2.2.6 Optimization Techniques	14
2.2.7 Genetic Algorithm	14
2.3 Technologies	16
2.3.1 Development Tools	16
2.3.1.1 HTML	16
2.3.1.2 JavaScript	16
2.3.1.3 TypeScript	17
2.3.1.4 Node.js	17
2.3.1.5 React	17

2.3.1.6 Python	17
2.3.1.7 JupyterLab	18
2.3.1.8 Figma	18
2.3.2 HTTP	18
2.3.3 Web Application Framework	19
2.3.3.1 Express	19
2.3.3.2 Flask	19
2.3.4 Database Tools	20
2.3.4.1 NoSQL	20
2.3.4.2 MongoDB	20
2.3.4.3 Mongoose	20
2.3.5 Deployment Tools	21
2.3.5.1 Google Cloud App Engine	21
2.3.5.2 MongoDB Atlas	21
2.3.6 External Service	21
2.3.6.1 Facebook Graph API	21
2.3.6.2 Google Cloud Storage	21
2.4 Related Research/Competing Solutions	22
2.4.1 Entrée	22
2.4.2 Eatsee	22
2.4.3 Whatever Mans	23
2.4.4 Tinder	24
2.4.5 On the design of Individual and Group Recommender Systems for Tourism Paper	24
2.4.6 Dynamic Music Recommender System Using Genetic Algorithm	25
3. METHODOLOGY AND DESIGN	26
3.1 System Functionality	26
3.1.1 Requirements	26
3.1.2 Use Case Diagram	27
3.1.3 Use Case Narrative	28

3.1.3.1 Login	28
3.1.3.2 Register	28
3.1.3.3 Register	28
3.1.3.4 Individual Recommendation	29
3.1.3.5 Group Recommendation	30
3.1.3.6 Edit the Food Preference	30
3.1.3.7 See the Favorite List	30
3.1.3.8 Logout	31
3.1.4 Functional Breakdown Structure	32
3.2 System Architecture	33
3.2.1 System Overview	33
3.2.2 Restaurant Recommender System Architect	34
3.2.2.1 App Server Architecture	35
3.2.2.2 App Client Architecture	36
3.2.2.3 Recommender Architecture	36
3.2.2.4 Data Service Architecture	37
3.2.2.5 Database Architecture	38
3.3 User Journey	39
3.3.1 Activity Diagrams	39
3.3.1.1 Registration	39
3.3.1.2 Individual Restaurant Recommendation	40
3.3.1.3 Registration	41
3.3.2 Sequence Diagrams	42
3.3.2.1 Registration	42
3.3.2.2 Individual Restaurant Recommendation	43
3.3.2.3 Create Group	45
3.3.2.4 Group Restaurant Recommendation	46
3.4 User Interface Design	47
3.4.1 Login Page	48
3.4.2 Registration Page	49

3.4.3	Individual Restaurant Recommendation Page	51
3.4.4	Group Restaurant Recommendation Page	54
3.4.5	Profile Page	57
3.5	Database Structure	57
3.5.1	Collections	58
3.5.2	Schema	58
3.5.3	Indexing	59
3.6	Restaurant Recommendation Algorithm	60
3.7	Production Testing Plan	60
3.7.1	Production Deployment	60
3.7.2	User Evaluation	60
3.7.3	Application Evaluation	60
3.8	Recommender Systems	60
3.9	Text Processing Algorithms	61
3.9.1	Algorithm I	61
3.9.2	Algorithm II	61
3.9.2.1	Algorithm SUB	61
3.10	Development Tools	61
4. PROPOSED WORK		62
4.1	System Architecture	62
4.2	System Specifications and Requirements	62
4.3	Hardware Module 1	62
4.3.1	Component 1	62
4.3.2	Logical Circuit Diagram	62
4.4	Hardware Module 2	62
4.4.1	Component 1	62
4.4.2	Component 2	62
4.5	Path Finding Algorithm	62
4.6	Database Design	62

4.7	GUI Design	62
5. IMPLEMENTATION RESULTS		63
6. CONCLUSIONS		64
6.1	Problems and Solutions	64
6.2	Future Works	64
REFERENCES		65
APPENDIX		68
A	First appendix title	69
B	Second appendix title	71

LIST OF TABLES

TABLE	PAGE
2.1 Characteristics of Explicit and Implicit Feedback	10
2.2 Common HTTP Verbs	19
2.3 NoSQL Types and Description	20
2.4 Mongoose' Schema Types	21
3.1 Database Collection and Description	58
3.2 Restaurant Schema	58
3.3 User Schema	58
3.4 Recommendation Schema	59
3.5 Category Schema	59
3.6 Favorite Schema	59
3.7 test table method1	61
4.1 test table x1	62

LIST OF FIGURES

FIGURE	PAGE
1.1 Gantt Chart for the First Semester	6
1.2 Gantt Chart for the Second Semester	7
2.1 Collaborative Filtering Process	12
2.2 Communication Through an HTTP Protocol	18
2.3 Screenshot of Entrée Application	22
2.4 Screenshot of Eatsee Application	23
2.5 Screenshot of Whatever Mans Application	23
2.6 Screenshot of Tinder Application	24
2.7 GR SK Architecture from the Paper	24
3.1 Use Case Diagram	27
3.2 Functional Breakdown Structure of the System	32
3.3 System Overview	33
3.4 System Architecture	34
3.5 App Server Architecture	35
3.6 App Client Architecture	36
3.7 Recommender Architecture	37
3.8 Data Service Architecture	37
3.9 App Server Architecture	38
3.10 Registration Activity Diagram	39
3.11 Individual Restaurant Recommendation Activity Diagram	40
3.12 Group Restaurant Recommendation Activity Diagram	41
3.13 Registration Sequence Diagram	42
3.14 Individual Restaurant Recommendation Sequence Diagram	43
3.15 Create Group Sequence Diagram	45
3.16 Create Group Sequence Diagram	46
3.17 Start Page User Interface Design	48
3.18 Registration Page User Interface Design	49

3.19 Profile Setup Page User Interface Design	50
3.20 Home Page User Interface Design	51
3.21 Filter Window User Interface Design	52
3.22 Recommendation Success Page User Interface Design	53
3.23 Group Joining User Interface Design	54
3.24 Group Confirmation Page User Interface Design	55
3.25 Group Recommendation Page User Interface Design	56
3.26 Profile Management Page User Interface Design	57
3.27 Restaurant Collection's Indexes	59
3.28 User Collection's Indexes	60
3.29 The network model	61

LIST OF SYMBOLS

SYMBOL		UNIT
α	Test variable	m^2
λ	Interarrival rate	jobs/ second
μ	Service rate	jobs/ second

LIST OF TECHNICAL VOCABULARY AND ABBREVIATIONS

ABC	=	Adaptive Bandwidth Control
MANET	=	Mobile Ad Hoc Network

CHAPTER 1 INTRODUCTION

1.1 Keywords

Genetic Algorithm, Optimization Problem, Decision Making Process, Group Recommendation Systems, Web Application, Restaurants

1.2 Problem Statement, Motivation, and Potential Benefits

“Kin-Rai-Dee? (Where are we going to eat?)” is the most common phrase we ask ourselves and others almost every day. We also believe that many people will as well ask this simple question regularly around mealtimes whether they are going to eat alone or with their group. To decide what and where to eat can take more time than necessary which can lead to frustration and sometimes, in the end, the decision cannot be made.

From the problems online survey, it revealed that 62% of the respondents, 93 out of 148, felt restaurant selection in each meal is their main problem which up to 65.5 % faced the pain about restaurant selection in a group by occurring frequently in a group of friends. The major issue comes from a variety of factors and conditions which lead to the inability to decide. From further inquiries about how they solved the problem, we found that very few people will find a restaurant that matches the conditions of their group. They usually select the restaurant according to the majority which cannot serve everyone’s satisfaction. Some of them use the recommendation application to be their alternative way to decide. Moreover, in the worst case, they just eat separately.

Even though various applications can suggest food to users, almost all of them just randomize the suggestions without any logic. Moreover, many of them suggest only food, not a restaurant. Based on our research, some applications that can suggest a restaurant based on their users’ preference still cannot support a suggestion for a group.

In this project, we will implement a web application that can solve the mentioned problems and improve the existing solutions. Our application will recommend a restaurant based on individual or group preferences, behaviors, and customizations. We hope that the application will be able to reduce time-wasting in choosing restaurants and give us no more frustration.

1.3 Project Type

Potential Commercial Product

1.4 Proposed Method

1.4.1 Approach

From the problem statement above, we will develop a web application that suggests a restaurant that matches customer profiles. We will provide a diversity of restaurants in Bangkok from an external source and their gen-

eral information that facilitate their restaurant consideration. Also, we will suggest a restaurant that matches multiple preferences (multiple users) by multi-objective optimization algorithms.

1.4.1.1 Literature Review and Survey

Explore existing works and knowledge related to our project e.g. recommendation system, multi-objective optimization algorithms, machine learning model, software engineering for web application development, etc. Moreover, we survey and construct a customer journey.

1.4.1.2 Data Collection

Gather a restaurant profile from an external source which is the restaurant's Facebook pages. and user profile from our data collection tool.

1.4.1.3 Data Preparation

A preprocessing method includes cleaning, selecting, normalizing, transforming, etc. to be a training set.

1.4.1.4 Construct the Model

Design and construct a model and train with prepared data, model evaluation, and hyperparameter tuning. This model will be integrated with our backend service.

1.4.1.5 Design User Interface

Design an user interface for our web application. This design process includes prototype implementation and prototype testing which can improve the user experience.

1.4.1.6 Build a Web Application

Implement a web application from our design. This implementation includes frontend and backend development.

1.4.1.7 Validation

Validate our completed web application which integrated our model. This process includes feedback gathering from users, feedback evaluation, model performance evaluation, and application performance monitoring.

1.4.2 Objectives

- To collect and analyze the customer behavioral data.
- To implement an analytical model that can suggest the right restaurant to individuals or a group of people based on their preferences, profiles, and behaviors.

- To implement a web application that allows users and groups of users to input their preferences and obtain the recommended restaurants.
- To reduce time-wasting on a restaurant consideration.

1.4.3 Scope

- Our web application supports modern web browsers such as Google Chrome, Safari, Firefox.
- Our web application focuses on the person or group of people who eat in the restaurant (not focus on delivery).
- Internet access is required in order to use our application.
- Restaurants in our database obtained from an external source are only restaurants in Bangkok.

1.5 Original Engineering Content

In this project, we will develop the following software, dataset, and algorithm:

1.5.1 Web Application

A web application for users that suggests the right restaurant to the right people based on individual preferences and group preferences. This application also collects customer behavior data as well as feedback from users.

1.5.2 Customer Behavior Dataset

Customer behavior data will be collected throughout semesters by both our data collection tool which will be used before our final product and our web application. The data include the restaurant that users choose to go with time series and their demographic information.

1.5.3 Group Recommendation Algorithm

Our system will recommend restaurants to individuals or groups by using our recommendation algorithm which includes machine learning and multi-objective optimization algorithms.

1.6 Task Breakdown and Draft Schedule

1.6.1 Semester 1

1. Requirement Gathering
 - 1.1. Kick-off meeting
 - 1.2. Brainstorm and analysis
 - 1.3. Survey

- 1.3..1. Competitor survey
- 1.3..2. Paper survey
- 1.3..3. Target user survey
- 1.3..4. Market survey
- 1.4. Survey conclusion
- 1.5. Requirement conclusion
- 2. Create Proposal
- 3. Researching
 - 3.1. Recommendation algorithm researching
 - 3.2. Optimization algorithm researching
- 4. Midterm Project Presentation (Slide and video presentation)
- 5. Data Collection Phase I
 - 5.1. Restaurant profile collection from sources
 - 5.2. Implement user data gathering tool
 - 5.3. User profile collection from tool
- 6. Design
 - 6.1. UX/UI Design
 - 6.1..1. Customer journey
 - 6.1..2. Create wireframe
 - 6.1..3. Test wireframe
 - 6.1..4. UI design
 - 6.1..5. Prototype implementation
 - 6.1..6. Test prototype
 - 6.1..7. Improvement
 - 6.2. Architecture Design
 - 6.2..1. System architecture
 - 6.2..2. Database
- 7. Create Final Report
- 8. Final Project Presentation (Slide and video presentation)

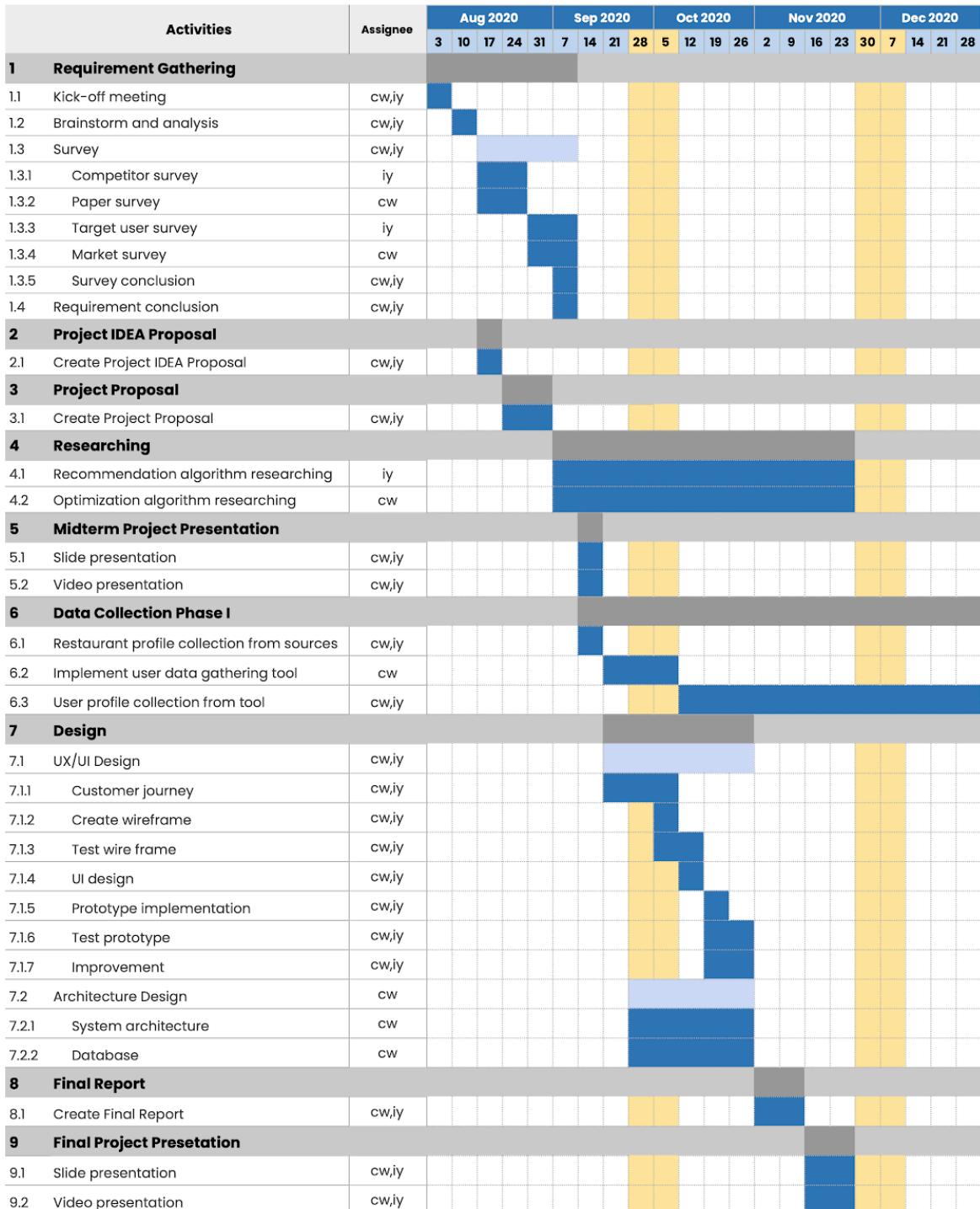
1.6.2 Semester 2

- 1. Development
 - 1.1. Application Development
 - 1.1..1. Web application implementation
 - 1.1..2. Testing

- 1.1..3. Application improvement
- 1.2. Model Development
 - 1.2..1. Model construction and training
 - 1.2..2. Model evaluation
 - 1.2..3. Model improvement
- 1.3. Integration Testing
 - 1.3..1. Testing
 - 1.3..2. Debugging
- 1.4. User Acceptance Testing
 - 1.4..1. User testing
 - 1.4..2. User feedback gathering
 - 1.4..3. Feedback evaluation
 - 1.4..4. System improvement
- 1.5. Create Report and Presentation
- 1.6. Production Deployment
 - 1.6..1. Backend deployment
 - 1.6..2. Frontend deployment
 - 1.6..3. Production testing

Semester 1 Gantt Chart

Web Application that Suggests the Right Restaurant to the Right People.

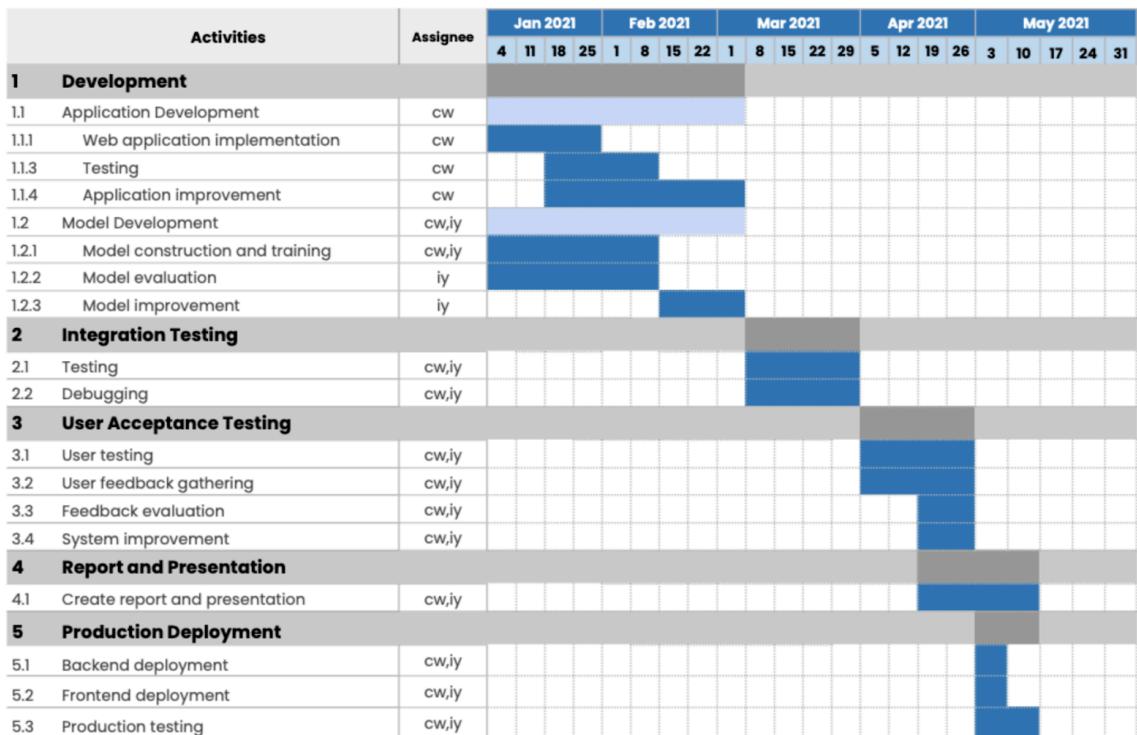


cw = Chanchana Wicha iy = Irin Yooktajarong

Figure 1.1 Gantt Chart for the First Semester

Semester 2 Gantt Chart

Web Application that Suggests the Right Restaurant to the Right People.



cw = Chanchana Wicha iy = Irin Yooktajarong

Figure 1.2 Gantt Chart for the Second Semester

1.7 Deliverables for Term 1

- Requirement Specification
- Survey Result
- Data Collection Tool
- Dataset (Customer Behavior and Restaurant Information)
- Customer Journey
- Prototype (Web Application)
- Database Schema
- System Architecture and Class Diagram
- Suggestion Algorithm and Model

1.8 Deliverables for Term 2

- Web Application (Frontend and Backend)

- Recommendation System both Individuals and Groups
- Feedback from Users
- Application Performance Evaluation
- Model Performance Evaluation

CHAPTER 2 BACKGROUND THEORY AND RELATED WORK

2.1 Background

As the world becomes more digital and technology is all around us, before we realized it, we were accustomed to a personalized experience. Many online businesses are implementing personalized recommendation systems that try to identify a user's preferences and provide them with relevant items to enhance the user's experience. However, mainstream restaurant recommendation applications in Thailand have not implemented personalized recommenders yet. This leads to the issue of finding an ideal restaurant which is always struggling for users and a person who is looking for new restaurants. Furthermore, the online problem survey revealed that many people faced a harder time to decide due to the group eating. Therefore, group decision making is very complex and time-consuming.

We brought this problem up to solve it since it is a common problem that everyone faces in daily life and no solution can be solved sustainably. Moreover, the majority of recommender systems are designed for individual users and the group recommender is challenging and new in the market. Inspired by such a challenge, we will invent a restaurant recommender system that could potentially reduce decision-making time for both individuals and groups and be an alternative way for users to opt for their ideal restaurants. Our system considers the interaction between users and restaurants and also uses side information e.g. users' tastes and preferences, restaurant profile, etc. This approach is able to maximize users' satisfaction and increase users' experience.

2.2 Related Theory

2.2.1 Web Application

A web application is an application program that is stored on a remote server and delivered over the internet through a browser interface. [1] Because web applications are delivered through the internet, users do not need to install the application and it also allows multiple users access to the application at the same time on the same version. Moreover, running web applications require only an internet browser, so a web application is a platform and operating system independent.

Because our platform needs to be easily accessible for users not only individual users but also every member in the group of users, having a web application that does not require any installation and platform independence is a perfect solution. Users can easily access our platform via their mobile internet browser and all of them will see the same version of the application. A web application also accommodates the development process since we do not need to implement the different versions for different platforms.

2.2.2 Information Collection

This collects relevant information of users to generate a user profile or model for the prediction tasks including the user's attributes, behaviors, or content of the resources the user accesses. [2] We need to do this process due to the recommender cannot provide results accurately until the users' profile and preferences have been

constructed. So, there are several types of information/feedback that can be captured and studied including explicit and implicit feedback.

2.2.2.1 Explicit Feedback

The accuracy of the recommendation depends on the number of ratings provided by the user. The only shortcoming of this method is, it requires effort from the users and also, users are not always ready to supply enough information. Despite the fact that explicit feedback requires more effort from the user, it is still seen as providing more reliable data, since it does not involve extracting preferences from actions, and it also provides transparency into the recommendation process that results in a slightly higher perceived recommendation quality and more confidence in the recommendations. [3]

2.2.2.2 Implicit Feedback

The system automatically infers the user's preferences by monitoring the different actions of users such as the history of purchases, navigation history, and time spent on some web pages, links followed by the user, content of an email, and button clicks among others. Implicit feedback reduces the burden on users by inferring their user's preferences from their behavior with the system. The method though does not require effort from the user, but it is less accurate. Also, it has been argued that implicit preference data might in actuality be more objective, as there is no bias arising from users responding in a socially desirable way. [3]

Table 2.1 Characteristics of Explicit and Implicit Feedback

Implicit Feedback	Explicit Feedback	Accuracy
Low	High	Abundance
High	Low	Context-sensitive
Yes	Yes	Expressivity of user preference
Positive	Positive and Negative	Measurement reference
Relative	Absolute	

Source: <https://core.ac.uk/download/pdf/207051652.pdf>

From Table 2.1, the accuracy of implicit feedback is lower than explicit feedback due to system monitoring only (indirect way). But with this approach, we can get abundant feedback whereas explicit feedback is scarce. Moreover, implicit feedback can capture only positive e.g. if a user never visits restaurant A that does not mean he/she does not like that restaurant. Although implicit feedback tends to be relative such as if a user visited restaurant B 10 times and restaurant C 100 times, then we can conclude that he/she has a higher preference for C than B. Lastly, as we can see from the table, one thing that is similar to each other is both are sensitive to the context

In this project, the PDPA or Personal Data Protection Act is not related to us. This rule mentioned about the protection of personal information. Any app that needs access to that data needs consent from the owner by informing the purpose of what information is collected and using that information for what purpose. However, we collected user information including username, password, and user profile: food preference where all of them do not relate to personal information.

2.2.3 Overview of Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems' ability to automatically learn and improve from experience without being explicitly programmed. [2] Machine learning is often categorized into 3 types which are supervised learning, unsupervised learning, and reinforcement learning.

In supervised learning, it is learned from the labeled data and predicts the future events from the labeled e.g. classified cats and dogs. In contrast, unsupervised learning deals with unlabeled data which is extremely useful to discover patterns, not just predict the future from labeled data e.g. customer segmentation. Last is reinforcement learning which is the closest to human learning by learning to take corrective action with the provided environment, for example, AlphaGo which lets the machine learn how to play Go.

As our project is a recommendation system that needs to predict the next user-preferred restaurants from interactions and side information which mainly are unlabelled data. So, the model works on its own to discover patterns and information which is the restaurant for each user based on preferences. In addition, this project is an unsupervised learning problem.

2.2.4 Recommendation System Techniques

Recommender systems are usually as if a decision making strategy that helps users select the interest items based on users' preferences or profile from overwhelmed available items or complex environments. There are 3 main types of recommendation filtering techniques which are content-based, collaborative, and hybrid. Each of these has different features and potentials.

2.2.4.1 Content-based Filtering Technique

Content-based filtering is a technique which recommends based on the similarity between the items and using users' profile/description. Items that are mostly related to the positive user's evaluation are recommended to the user. This approach uses a variety of models to find the similarity document to get meaningful suggestions by treating the recommendation as a user-specific classification problem by classifying for the user's likes and dislikes based on an item's features. [4] Moreover, it does not require other users' profiles as they do not affect recommendations, for example, the system wants to recommend the product which is similar to the product that user has purchased in the past, so they use content-based filtering to be the recommender by using the characteristics of the product to recommend including genre, price, style, description, picture, etc.

Therefore, this technique has the ability to recommend new items even if there is no user's evaluation/rating. This would mean that it is able to recommend accurately without user preferences which consequence that if there is a poor description of items and an unorganized user profile, it can lead to the worst recommendation. So, the effectiveness of the content-based filtering technique depends on the richness of descriptive information, but lastly, it is unable to recommend a product that is very different from the product that the user previously purchased, resulting in the user receiving a not diverse category of product. To overcome this, it can adapt to some form of a hybrid system.

Moreover, from this approach, our project can use the algorithm to be part of our recommendation system which makes the system have an ability to recommend a restaurant even if there's no user preference or we can say that we are able to overcome the new user problem.

2.2.4.2 Collaborative Filtering Technique

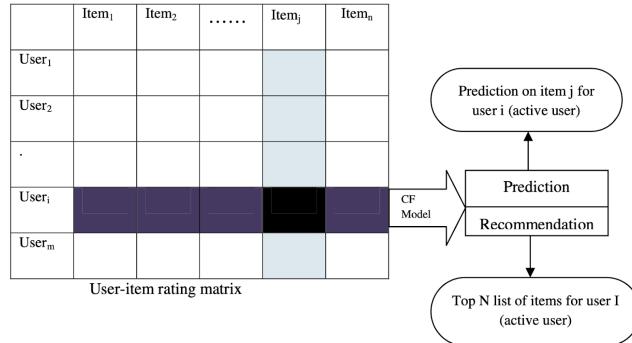


Figure 2.1 Collaborative Filtering Process

Source: <https://www.sciencedirect.com/science/article/pii/S1110866515000341>

A collaborative filtering technique as opposed to the content-based filtering technique which collaborative filtering cannot recommend the item by using only the description of metadata. The key idea of this technique is a group of people with similar tastes, often tend to like similar items. So, collaborative filtering uses only data about rating profiles from different users or items in order to generate the recommendation. Here is the flow of this technique:

1. Gather user expression from her/his preferences by items rated.
2. Match the user's ratings with other users' then find the most similar tastes.
3. Recommend with the high rating of unseen items from similar users to this user.

To recommend, it works by creating a user-item matrix of preferences for items by users, as Figure 2.2.4.2 above which represents the process start from the matrix to the output of this system, and then calculating the similarity between profiles to get the relevant item that matches the user. Thus, the user will be able to get the items that he/she has not rated yet but already rated from a similar user. This approach can produce the output either prediction or recommendation by prediction represented as a score of an item for the user whereas recommendation is represented as a top N list of items as mentioned in Figure 2.2.4.2. Collaborative filtering techniques can separate into 2 main categories: memory-based and model-based.

2.2.4.2.1 Memory-based

Memory-based can approach in 2 ways which are user-based and item-based.

User-based collaborative filtering is to find the users who are similar to the target user and then recommend by calculating the weighted averages of the ratings from the group of similar users. For item-based collaborative filtering is calculating the similarity between items instead of users for determining the items which are similar to the target item then, calculating the weighted average of the ratings from a group of similar items.

To calculate the weighted average, there are several popular similarity measurements to find the similarity between user/item including Pearson correlation coefficient and Cosine similarity.

$$simil(x, y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r}_x)(r_{y,i} - \bar{r}_y)}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r}_x)^2} \sqrt{\sum_{i \in I_{xy}} (r_{y,i} - \bar{r}_y)^2}} \quad (2.1)$$

From the above example equation 2.1, it is a Pearson correlation coefficient, $simil(x, y)$ denotes the similarity between 2 users x and y , mean the rating given to the item i by user x , \bar{r}_x means the mean rating of user x , I_{xy} and is equal to a total number of items in the user-item.

$$\cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \times \|\vec{y}\|} = \frac{\sum_{i \in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum_{i \in I_{xy}} r_{x,i}^2} \sqrt{\sum_{i \in I_{xy}} r_{y,i}^2}} \quad (2.2)$$

For cosine similarity 2.2, it is different from Pearson correlation coefficient which calculates as a vector-space model by measuring the similarity between dimensional vectors based on the angle which $\cos(\vec{x}, \vec{y})$ denotes to the similarity between 2 users x and y .

2.2.4.2.2 Model-based

This approach is able to improve the performance of collaborative filtering by constructing the model which is done by using machine learning or data mining techniques. Moreover, with this approach, dimensionality reduction methods are mostly being used as complementary techniques to improve the robustness and accuracy of the memory-based approach. [3]

For example, this technique is matrix factorization which is able to discover hidden concepts and their relationship with users and items.

The advantage of the collaborative filtering technique is performed in domains where there is not much content associated with items and where content is difficult for a computer system to analyze such as opinions and ideas. [3] But there is some disadvantage which is about the cold-start problem which refers to the new user or item that comes with the empty profile since the user has not rated any item yet. Moreover, a memory-based approach has a problem with scalability. This issue can be solved by the model-based approach.

Collaborative filtering can perform better than content-based filtering in poor content. With the advantages of this technique, we are able to reduce the user effort by performing with the implicit feedback that comes from the system monitoring and get the recommendation result more accurately.

2.2.4.3 Hybrid Filtering Technique

The hybrid filtering technique is the most recommender system used nowadays. It is a combination of content-filtering, collaborative filtering, and other approaches. There are several ways to implement e.g. doing content filtering and collaborative filtering predictions separately and then combining them later, etc. Combining different recommendation techniques can provide a better system and be able to overcome the limitations of a pure recommendation system.

Therefore, the Hybrid approach can give a better recommendation than a pure approach and can avoid the cold-start problem, sparsity, scalability, and other problems.

According to both techniques which are content-based and collaborative filtering, we found that they can reduce both of their problems whether cold-start problems that come from collaborative filtering which are content-based able to overcome, or about the diversity recommendation problem which hybrid approach can overcome. In our opinion, the hybrid filtering technique is the best choice to be done in a restaurant recommendation system.

2.2.5 Aggregation Mechanism

Since this project aims to recommend the restaurant to users both individuals and groups, we need to aggregate individual preferences into group preferences before recommending to a group of users. There are many existing strategies for aggregation but the key idea of this approach is how the individual group members' preferences are combined into a group preference.

For the aggregating preferences strategy, the individual preferences will be aggregated into groups by combining the user's preferences/ratings of each item and let this be the group rating. Then, group recommendations will be based on this preference and offer only the top N recommendations list to the group which N refers to the number of items that will be recommended to a group.

2.2.6 Optimization Techniques

The optimization algorithm is a procedure which is executed iteratively by comparing various solutions till an optimum or a satisfactory solution is found [5] which can be either the minimum or the maximum value. The optimal value can be selected from a set of available solutions with some criterion [6]. Examples of objective functions that require maximum values are: number of kilometers that a delivery vehicle can run per 1 gallon of fuel, number of customers, number of customers eating in a restaurant per hour, etc. Examples of objective functions that require minimum values are: cost per unit of production, factory electricity consumption per hour, and so on.

In this project, we have adopted a method of optimization to define the objective function of our system. The objective of our system is to maximize the satisfaction of users based on the recommendations of restaurants that meet the interests and needs of that user.

2.2.7 Genetic Algorithm

The genetic algorithm (GA) is a metaheuristic method inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on biologically inspired operators such as mutation, crossover, and selection. [7]

In the genetic algorithm, the optimal solution is found by replacing an existing set of parameters in the form of chromosomes. Each parameter in the chromosome is known as a gene which is usually replaced with binary numbers by encoding methods. Encoding of chromosomes is the first step in solving the problem and it depends on the problem. The initial population will be random values. To find the fitness solution, each generation will either mutate or switch genes between each other until a new generation of population with

more fitness is obtained, this evolving process will continue until a suitable answer is found. The overall process of genetic algorithms is shown in the following pseudocode.

Genetic Algorithm Pseudocode[8]

- 1** START
- 2** Generate the initial population
- 3** Compute fitness
- 4** REPEAT
- 5** Parent Selection
- 6** Crossover/Mutation
- 7** Compute fitness
- 8** UNTIL population has converged
- 9** STOP

The genetic algorithm consists of 4 main phases: Initial Population, Selection, Reproduction, and Termination. The details of each phase are as follows.

1. Initial Population

Before getting the population, the problem needs to go through the encoding methods which can be binary, permutation, or value encoding. Binary encoding is the most common method that represents the solution in strings of 1s and 0s. Permutation encoding is usually used for Travelling Salesman Problem which every chromosome is a string of numbers. Value encoding is used in the problem where complicated values and binary method would not be sufficient, so that it is necessary to develop a specific crossover and mutation method.

Population comes from the set of chromosomes which each chromosome is a solution to a problem that needs to be solved. Genes can combine to be a chromosome which is known as the solution. The value of genes is randomized, with the initial number of genes dependent on the problem and sometimes can be significantly randomized to get closer to the answer.

2. Selection

The idea of selection is to select the appropriate chromosomes or the fitter chromosomes measured by a fitness function and pass them to the next generation to be able to get a more optimal answer to the problem. Generally, chromosomes with higher fitness have more chances than lower fitness to reproduction.

3. Reproduction

After the selection process, this process will use those chromosomes to generate the next generation by either crossover or mutation. Crossover is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a crossover point is chosen at random from within the genes. [8] After that, offspring will be created and added to the population of next generation. For mutation, it occurs when new offspring are

formed by flipped some bits. This process exists to maintain the diversity of population and prevent premature convergence.

4. Termination [7]

This generational process is repeated until a termination condition has been reached. Common terminating conditions are:

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above

In this approach, we can create a recommendation system for both individuals and groups which is based on users' preferences along with the user's restaurant selection history. Genetic algorithms approach is the most likely to be able to serve our objective which is to maximize the satisfaction of users.

2.3 Technologies

2.3.1 Development Tools

2.3.1.1 HTML

HTML (HyperText Markup Language) is the most basic building block of the Web. It defines the meaning and structure of web content. HTML uses "markup" to annotate text, images, and other content for display in a Web browser. HTML markup includes special "elements" such as `<body>`, `<p>`, `<div>` and many others. An HTML element is set off from other text in a document by "tags", which consist of the element name surrounded by "<" and ">". [9]

2.3.1.2 JavaScript

MDN [10] states that JavaScript is a programming language that enables complex feature implementation on web pages such as displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc. Many modern browsers such as Google Chrome, Safari, and Firefox support JavaScript and it can run immediately within the browsers without repeatedly calling the server.

The reason that JavaScript enables those complex feature implementation mentioned above is because it support the use of Document Object Model (DOM), a set of APIs for controlling HTML and styling information. [11] With DOM, we can write JavaScript to interactively manipulate the element in HTML easily.

Because our platform needs to serve an interactive user interface with a fast speed, these characteristics and advantages of JavaScript can fulfill those requirements. However, there are limitations of JavaScript which are reading and writing local files and browser interpretation. The first limitation, it lacks the feature of reading and writing local files for security reasons, which we need another server side service to handle the more complex tasks. The other one is that different browsers can sometimes interpret JavaScript code differently, but since we target to support most of the modern browsers, we do not have to worry about this limitation.

2.3.1.3 TypeScript

TypeScript extends JavaScript by adding types. By understanding JavaScript, TypeScript saves us time catching errors and providing fixes before we run code. [12] The reason that we choose TypeScript over plain JavaScript is it makes our platform easily maintainable by adding static types and error checking.

2.3.1.4 Node.js

Node.js is an open-source and cross-platform JavaScript runtime environment. [13] Because we choose to have a JavaScript web application framework, Node.js is essential for running our scripts. In the server side services, Node.js is the environment that will run the application and actively serves the request from clients while in client side services, the services might need Node.js to build the optimized bundled package which serves the client.

2.3.1.5 React

React is a library for building composable user interfaces. It encourages the creation of reusable UI components which present data that changes over time. [14] The DA14 blog [15] demonstrates the top 10 advantages of React including JSX syntax and reusable components. The first advantage, React uses JSX syntax which is an extension syntax to JavaScript. It makes subcomponent rendering easier by accepting HTML quoting, thus the overall process of writing components is much simpler. The other one of their best features is an ability to reuse components which makes a codebase more clean and easy to maintain. In our project, we want our development process to be simple and our codebase to be clean and maintainable, so React is the perfect solution. Moreover, React supports TypeScript that adds the static type to our JavaScript which also facilitates our development process.

2.3.1.6 Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. [16] According to PYPL PopularitY in November 2020, Python is the most popular programming language. Due to its popularity, Python has hundreds of different libraries and frameworks which is a great addition to your development process. [17] Many of these libraries and frameworks are mainly focused on data analytics and machine learning. Examples of those libraries are Numpy, Pandas, and Matplotlib. Our platform not only

focuses on user experience but also data analytic and machine learning, so using Python fits perfectly in those areas.

2.3.1.7 JupyterLab

JupyterLab enables you to work with documents and activities such as Jupyter notebooks, text editors, terminals, and custom components in a flexible, integrated, and extensible manner. [18] It is flexible and easy to arrange multiple notebooks or documents by showing in the tabs pattern. So, to do the experiment and construct the model of a recommendation system whether individuals or groups approach, this platform can provide a suitable working area to us. Moreover, JupyterLab also supports a variety of data formats e.g. images, CSV, JSON, etc.

2.3.1.8 Figma

Figma is a cloud-based design and prototyping tool for digital projects. It's made so that users can collaborate on projects and work pretty much anywhere. [19] We choose Figma because of their collaborative feature which allows us to work together easily. Figma also allows us to test our prototype by sharing an online link to our testers.

2.3.2 HTTP

HTTP is a protocol that allows the fetching of resources, such as HTML documents. It is the foundation of any data exchange on the Web and it is a client-server protocol, which means requests are initiated by the recipient, usually the Web browser. [20]

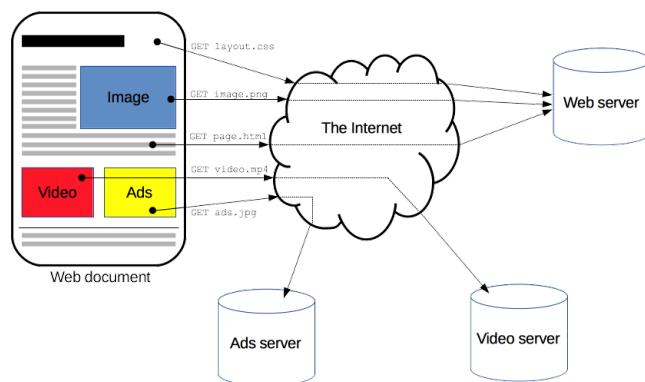


Figure 2.2 Communication Through an HTTP Protocol

Source: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>

Figure 2.3.2 shows how messages are communicated via the HTTP protocol. As you can see clients and servers communicate by exchanging individual messages. [20] The messages sent from clients are requests, and the reply messages from servers are called responses.

HTTP defines a set of request methods for requests to indicate the desired action to be performed for a given resource. Although they can also be nouns, these request methods are sometimes referred to as HTTP verbs. [21] The common HTTP verbs are shown in Table 2.2.

Table 2.2 Common HTTP Verbs

Verb	Description
GET	Retrieve data.
POST	Create data
PUT	Update data.
DELETE	Remove Data

One of the HTTP characteristics is HTTP is a stateless protocol in which no session information is retained by the receiver or server. Relevant session data is sent to the server by the client in such a way that every request sent can be understood in isolation, without context information from previous requests in the session. [22] To make servers manage and understand the session, we need to make use of Cookies which is compact-sized information sent with requests. The servers can use that information to identify the current session from every isolated request such as authentication which makes them understand who is sending the request and do not require users to login every time they need to interact with the system.

2.3.3 Web Application Framework

A web application framework is a software framework that includes a code library that makes web development quicker and easier by giving basic patterns for building reliable, scalable and maintainable web applications [23] so that we do not need to implement everything from the ground up. It can facilitate various functionalities for web development such as request handling, URL routing, transferring data phrasing and database connection configuration.

In our project, we choose Express framework for JavaScript with Node.js and Flask framework for Python. Both of them have a suitable functionality for our system with a large community base.

2.3.3.1 Express

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. [24] The main advantages of Express are scalability and the language. Express lets us scale our application easily because of the support of Node.js and its lightweight. Express also uses JavaScript with TypeScript support, so we do not need to learn another programming language as we considered using TypeScript for frontend development.

2.3.3.2 Flask

Flask is a lightweight web application framework for Python. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. [25] Unlike full-stack web application frameworks such as Django, Flask offers only essential features for building a core web application, thus the performance is better and we can keep our codebase clean and simple. We need Flask in addition to Express because our project also focuses on data analytics and machine learning which also need communication to other components and implemented with Python, so Flask can seamlessly serve us those purposes.

2.3.4 Database Tools

2.3.4.1 NoSQL

NoSQL databases are non-tabular and store data differently than relational tables. They provide flexible schemas and scale easily with large amounts of data and high user loads. NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph. [26] The description of each type is shown in Table 2.3.

Table 2.3 NoSQL Types and Description

Type	Description
Document	Store data in JSON-like objects, each field will have a key-value pair. Values in each field can have various types so that document-based NoSQL can be used as a general-purpose database. It also supports horizontal scaling to support a large volume of data.
Key-value	Key-value based NoSQL is simpler than document-based NoSQL because it contains one key and value. It is used for a large amount of data but not for a complex query.
Wide-column	Store each row of data in dynamic columns. It is suitable for data that we can predict the query pattern.
Graph	Store data in nodes and edges. Nodes generally store information while edges store their relation to other nodes.

According to Table 2.3, the advantages of NoSQL which is flexible schemas and document-based NoSQL type which support complex query and horizontal scalability, we can conclude that document-based NoSQL is suitable to be our database tool because our data's schema can easily change over time, we have a large amount of data and we need a complex query method for retrieving the data such as query objects sorted by distance from a coordinate.

2.3.4.2 MongoDB

MongoDB is a document-based NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB. Collections contain sets of documents and functions which are the equivalent of relational database tables. [27] We choose to use MongoDB because it is open-source software, large community based, powerful query language and natively support database cloud service. We consider using their native cloud platform, MongoDB Atlas, which will be described later, so that we do not have to deploy our own database service by ourselves and the server side services can easily connect to the database with a simpler configuration.

2.3.4.3 Mongoose

Mongoose is a JavaScript library which provides a straight-forward, schema-based solution to model your application data. It includes built-in type casting, validation, query building, business logic hooks, and more, out of the box. [28] It also provides an interface to connect to the database. Although NoSQL does not have a fixed schema, having a schema for some essential fields in the model helps in a uniform data format that facilitates database operations in our system. Mongoose provides essential types for declaring the model's schema. The essential types are shown and described in Table 2.4 below.

Table 2.4 Mongoose' Schema Types

Type	Description
ObjectId	A special type typically used for unique identifiers
String	A sequence of characters in string format
Number	A real number which includes all types of number such as integer, float, or double.
Boolean	True or false state
Date	Datetime format
Object	Nested object

2.3.5 Deployment Tools

2.3.5.1 Google Cloud App Engine

Google Cloud App Engine is a fully managed, serverless platform for developing and hosting web applications at scale. It lets you choose from several popular languages, libraries, and frameworks to develop your apps, then let Google Cloud App Engine take care of provisioning servers and scaling your app instances based on demand. [31] Google Cloud App Engine is suitable for our platform which has 2 primary programming languages. It makes the deployment process much more simple.

2.3.5.2 MongoDB Atlas

MongoDB Atlas is a fully-managed cloud database developed by the same people that build MongoDB. Atlas handles all the complexity of deploying, managing, and healing your deployments on the cloud service provider. [32] MongoDB Atlas is the perfect choice for us because it gives us life-time free 500 MB of storage space and all of the advantages of NoSQL stated before.

2.3.6 External Service

2.3.6.1 Facebook Graph API

Facebook Graph API is the primary way for apps to read and write to the Facebook social graph [33]. It provides a place query that we use for collecting restaurant data. Their restaurant information data includes restaurant tags which can help us in data analytics and machine learning process. We will write a script to query restaurant information from their API. Their API is free to use with the authentication token. The token is acquired by setting up the application on their platform.

2.3.6.2 Google Cloud Storage

Google Cloud Storage is a web service that allows storing and accessing data on a cloud. It provides a uniform process to access the static data. In our project, we need to keep the image of the restaurants and the total size of the images is too large to serve via our main server, Google Cloud Storage is the solution for this problem. Unlike Google Drive, Google Cloud Storage can set the static path for every restaurant image, but Google Drive will generate a random url to access the data. Another cloud storage service is Amazon S3, however, since we use Google Cloud Platform service which is Google Cloud App Engine for the deployment, Google Cloud Storage is a better choice for this project.

2.4 Related Research/Competing Solutions

2.4.1 Entrée

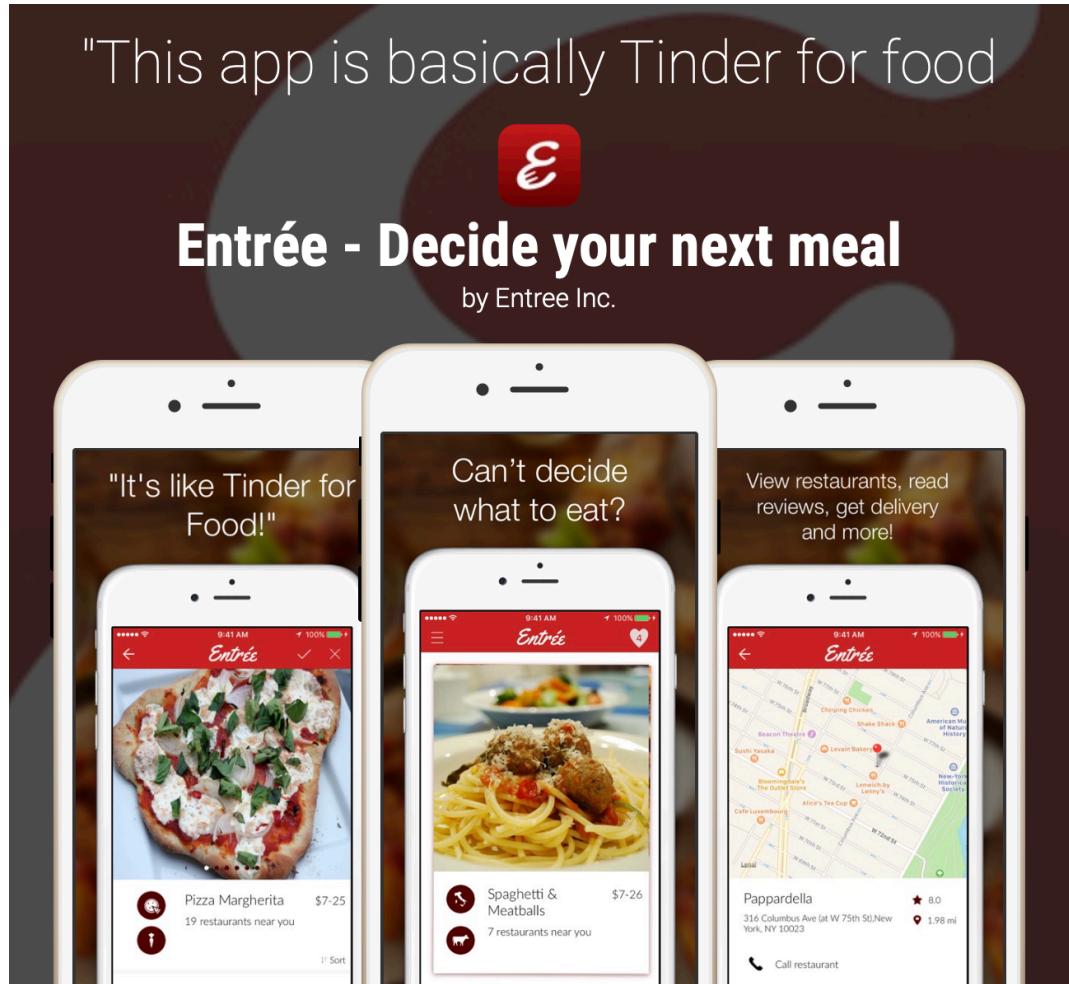


Figure 2.3 Screenshot of Entrée Application

Source: <https://appadvice.com/app/entr-c3-a9e-decide-your-next-meal/1037913795>

Entrée aims to solve the endless problem about what to order every day, by acting as a sort of “Tinder meets Pandora” for food. The app, freely available for download on iOS, shows users high-quality images of food or meals from nearby restaurants, based on an algorithm that learns users’ preferences. It also provides contextual recommendations, based on the occasion the user is looking to eat out for, like birthdays or date nights. The app is functional in 30 cities, and in the cities where Uber is also available, it allows users to book an Uber to the restaurant, call the restaurant and place your order for delivery. The app utilizes data from Foursquare for reviews and other information. [34]

Compared to our system, Entrée can suggest restaurants for only individuals, in contrast, our system support group suggestion. It supports 30 cities outside Thailand but our system has restaurants that are located in Thailand which uses Thai restaurant databases such as Wongnai and Google Places.

2.4.2 Eatsee

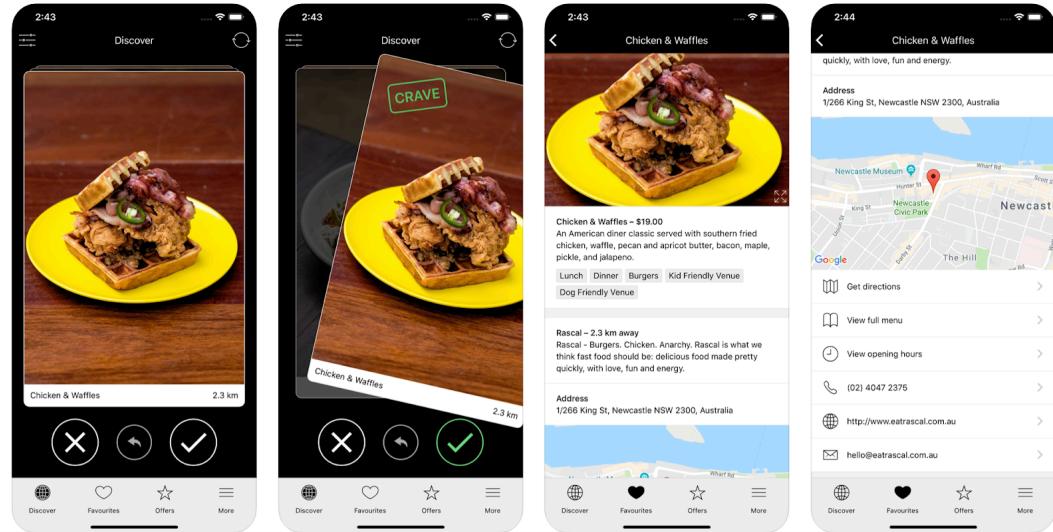


Figure 2.4 Screenshot of Eatsee Application

Source: <https://apps.apple.com/us/app/id1202110881>

Eatsee is an application that helps users find dishes that they'll love by setting their preferences and swipe through user-tailored results, tap to find out more about the restaurant, then swipe right to save it to be favorites. Eatsee is currently available in Newcastle (New South Wales, Australia), Coffs Harbour (New South Wales, Australia), and San Francisco (California, United States). [35]

Compared to our system, Eatsee only suggests random food based on their preference setting which doesn't learn about users' behavior. It focuses on food, not restaurants but our system prioritizes restaurants over food. It doesn't support group suggestions and restaurants located in Thailand. 2whatman

2.4.3 Whatever Mans



Figure 2.5 Screenshot of Whatever Mans Application

Source: <https://play.google.com/store/apps/details?id=com.DEC.Eatwhat>

Whatever Mans Application is a Thai food random application that has an interaction with the user by creating an interactive animation called Pa-Man to talk with the user. Whatever Mans is available for download on Android only.

Compared to our system, Whatever Mans Application supports only on an Android platform whereas our system supports both iOS and Android because of web application technology. Their suggestion isn't based on users' profiles or preferences which means their suggestion is just randomization. Although it supports Thai food, it can suggest only food, not restaurants. It also doesn't support group suggestions.

2.4.4 Tinder

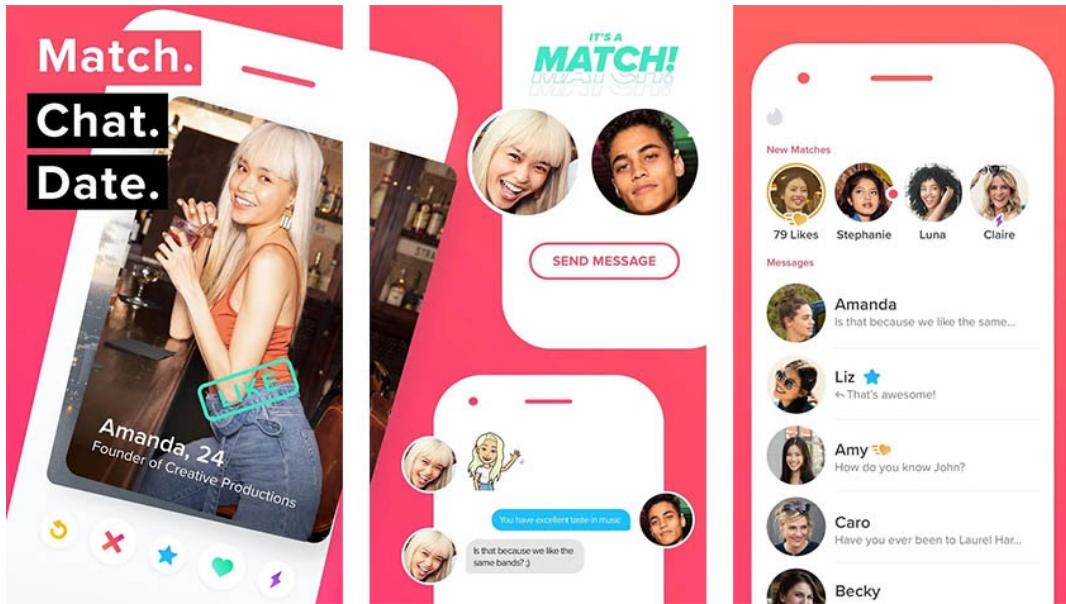


Figure 2.6 Screenshot of Tinder Application

Source: <https://www.androidauthority.com/best-tinder-alternatives-android-1084559/>

Tinder is a geosocial networking and online dating application that allows users to anonymously swipe to like or dislike other profiles based on their photos, a small bio, and common interests. Once two users have "matched", they can exchange messages. [36]

Compared to our system, Tinder shares a similarity with our system which is a recommendation algorithm. It learns user preferences over time and suggests new people based on their preferences. However, their suggestion is based on individuals because it is a dating application. The main contrast to our system is that Tinder focuses on suggesting new people whereas our system suggests restaurants.

2.4.5 On the design of Individual and Group Recommender Systems for Tourism Paper

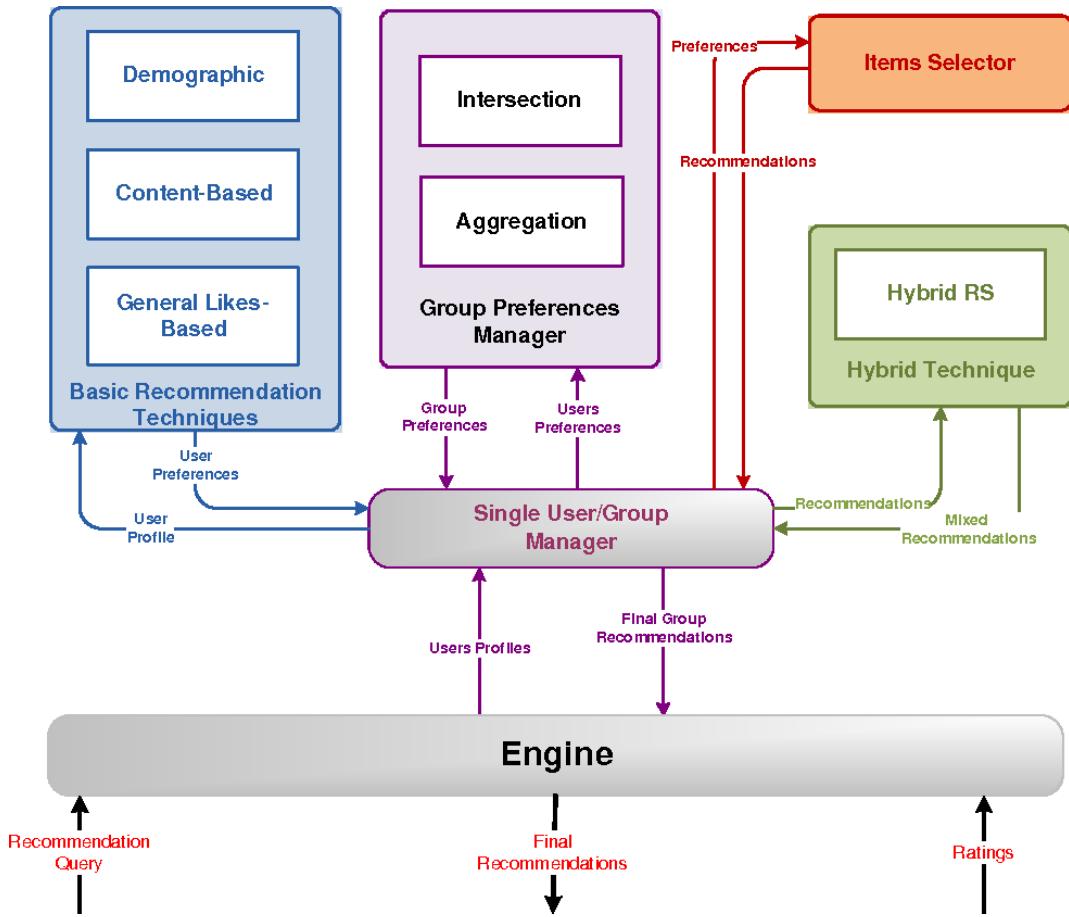


Figure 2.7 GRSK Architecture from the Paper

Source: <https://www.semanticscholar.org/paper/On-the-design-of-individual-and-group-recommender-Garcia-Sebastia/b367686e60db2f68102cd14b1360e9fc26a1757b>

This paper presents a recommender system for tourism based on the tastes of the users, their demographic classification, and the places they have visited on former trips. The system is able to offer recommendations for a single user or a group of users. The group recommendation is elicited out of the individual personal recommendations through the application of mechanisms such as aggregation and intersection. The elicitation mechanism is implemented as an extension of e-Tourism, a user-adapted tourism and leisure application whose main component is the Generalist Recommender System Kernel (GRSK), a domain-independent taxonomy-driven recommender system. [37]

Compared to our system, the algorithm designed in this paper can be adapted with our group recommendation system. However, their study focuses on tourism but our system focuses on restaurant recommendation.

2.4.6 Dynamic Music Recommender System Using Genetic Algorithm

This paper represents a music recommendation system that is able to recognize and provide items corresponding with user favorites. The system is able to identify the n-number of users' preferences and adaptively recommend music tracks according to user preferences by extracting unique features of each music track. Then

applying BLX-a crossover to the extracted features of each music track. [39] The recommender is base on genetic algorithm along with content-based filtering technique for generate the initial population in genetic algorithm which can divided into 3 phase including feature extraction phase, evolution phase, and interactive Genetic algorithm phase. User favorite and user profiles are included. After that the successive page is constructed based on the user evaluation.

Compared to our system, the information of this paper that uses the genetic algorithm is similar to our system which are user favorite and user profiles. But the differences are our objective function and the system architecture in which we focus on the restaurant.

CHAPTER 3 METHODOLOGY AND DESIGN

3.1 System Functionality

3.1.1 Requirements

- Users are able to register if they do not have an account.
- Users are able to logout from our system.
- An individual user is able to see the suggested restaurants.
- Users are able to set restaurant preferences for each meal.
- Users are able to see information about restaurants.
- Users are able to select the restaurant that they want to go from the individual recommendation.
- Users are able to rank the restaurants that they want to go by their preference from the group recommendation.
- Users are able to join an eating group using a group pin code.
- A group of users is able to start a group recommendation.
- Users are able to save restaurants to their favorites and view them later.
- Users are able to edit their food preferences.
- The system must recommend restaurants to individuals and groups based on their preferences and behavior.
- The system can suggest nearby restaurants based on users' current location.

3.1.2 Use Case Diagram

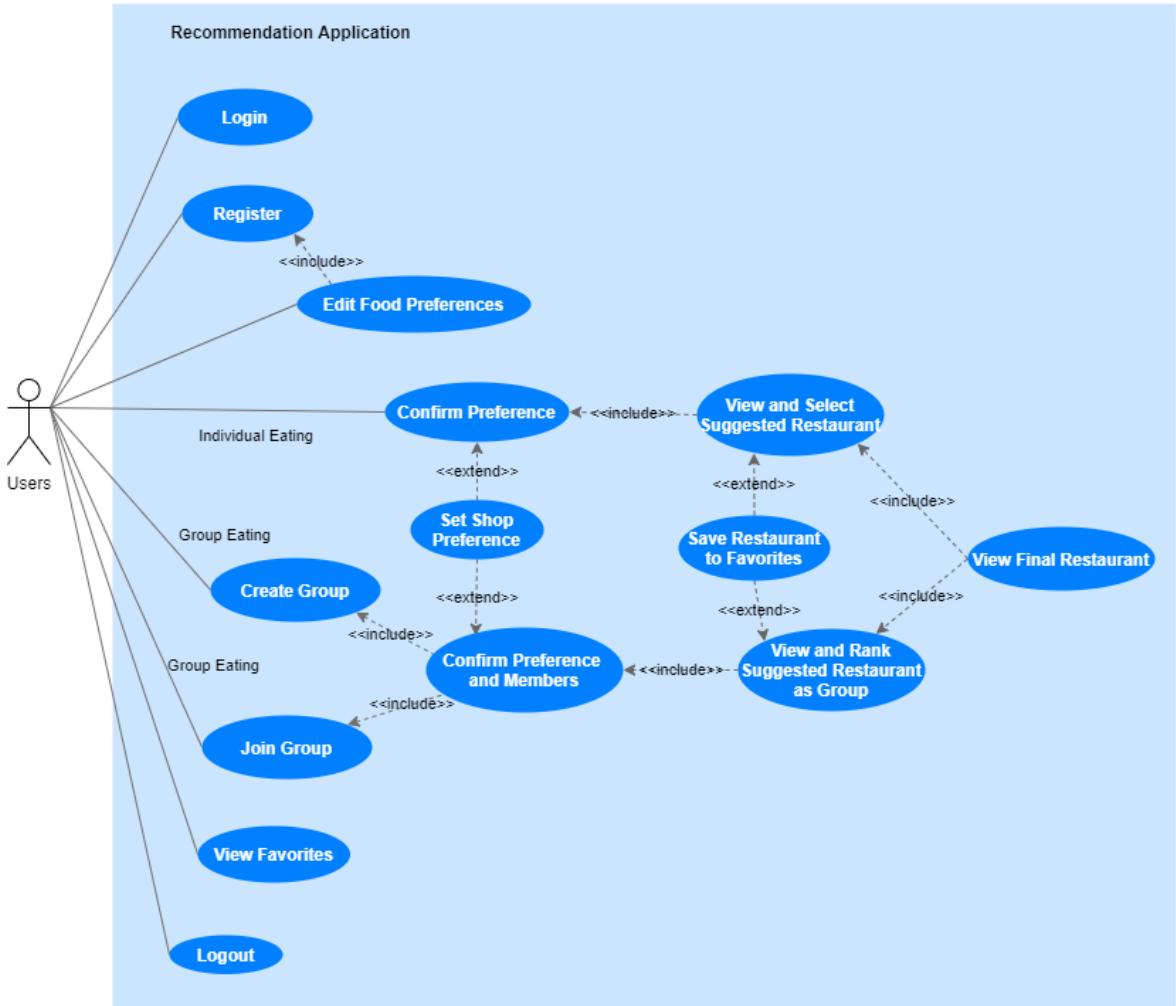


Figure 3.1 Use Case Diagram

1. Users can log in to the system using their username and password.
2. Users can register and users are required to set their food preference first before using the system.
3. Users can confirm the preference and see the suggested restaurants as individual eating and they can select the suggested restaurants. Moreover, users can save the restaurant into their favorite list.
4. Users can create a group or join other groups, then they can see the group confirmation page, and start the group recommendation.
5. Users can edit their food preference again.
6. Users can see their favorite list.
7. Users can log out from our system.

3.1.3 Use Case Narrative

3.1.3.1 Login

Name: Users Login

Actors: Users

Goal: Users want to login into the system.

Preconditions: None

Main Success Scenario:

1. Users input their valid username and password.
2. User is logged in.

Extensions (a)

- 1a. Input username or password is not correct.
- 2a. System asks a user to input username and password again.
- 3a. Go back to the main scenario number 1.

Extensions (b)

- 1b. Input username or password is not correct.
- 2b. Users go to a registration page.

Postconditions: None

3.1.3.2 Register

Name: Users Login

Actors: Users

Goal: Users want to login into the system.

Preconditions: None

Main Success Scenario:

1. Users input their valid username and password.
2. User is logged in.

Extensions (a)

- 1a. Input username or password is not correct.
- 2a. System asks a user to input username and password again.
- 3a. Go back to the main scenario number 1.

Extensions (b)

- 1b. Input username or password is not correct.
- 2b. Users go to a registration page.

Postconditions: None

3.1.3.3 Register

Name: Users Registration

Actors: Users

Goal: Users do not have an account and want to use the system.

Preconditions: User open registration page.

Main Success Scenario:

1. System will show a general information form to the user.
2. Users fill the valid username and password in the general form.
3. Users fill all general fields in the general form.
4. Users confirm the general form.
5. System will ask the user to fill a food preference form.
6. Users fill the food preference form.
7. Users confirm the registration form.
8. User is registered and logged in.

Extensions (a)

- 2a. Username is invalid, already existed or password confirmation is mismatched.
- 3a. System asks the user to input a valid username and password again.
- 4a. Go back to the main scenario number 2.

Extensions (b)

- 3b. User didn't fill some fields in the general form.
- 4b. Users confirm the general form.
- 5b. System asks the user to fill all the fields in the general form.
- 6b. Go back to the main scenario number 3.

Postconditions: None

3.1.3.4 Individual Recommendation

Name: Individual Recommendation

Actors: Users

Goal: Users want the system to suggest restaurants for them.

Preconditions: User is logged in and opens an individual recommendation page.

Main Success Scenario:

1. User allows the system to access their current location
2. The system shows a confirmation page to the user.
3. Users confirm the setting.
4. The system will show the suggested restaurant to the user
5. Users select the restaurant that they want to go to.
6. Users see their selected restaurant as a final restaurant.
7. Recommendation completed

Extensions (a)

- 1a. Users do not allow the system to access their current location.
- 2a. The nearby restaurant suggestion will be disabled.
- 3a. Continue to the main scenario number 2.

Extensions (b)

- 2b. Users change the setting for that meal.
- 3b. Go back to the main scenario number 3.

Extensions (c)

- 4c. Users save the restaurant to their favorite list.
- 5c. Go back to the main scenario number 5.

Postconditions: None

3.1.3.5 Group Recommendation

Name: Group Recommendation

Actors: Users

Goal: Users want the system to suggest restaurants for their group.

Preconditions: Users are logged in and open a group recommendation page.

Main Success Scenario:

1. Users allow the system to access their current location.
2. Users see the confirmation page and group pin code.
3. Users share pin code or QR code to others members.
4. Every member joined.
5. Head of the party selects to start the recommendation.
6. Everybody in the group sees the same set of the restaurants.
7. Everybody selects their restaurant they want to go to by their preference.
8. Everybody finished selecting the restaurants.
9. The system shows the final restaurant to the group.
9. Recommendation completed

Extensions (a)

- 3a. Users choose to join the group by inputting the group pin.
- 4a. Group pin is invalid.
- 5a. Users cannot join the invalid pin group.

Extension (b)

- 6b. Users save the restaurant to their favorite list.
- 7b. Continue to the main scenario number 7

Postconditions: None

3.1.3.6 Edit the Food Preference

Actors: Users

Goal: Users want to edit their food preference.

Preconditions: Users are logged in.

Main Success Scenario:

1. Users choose to edit their food preference.
2. The old food preference will be shown and let users edit them.
5. User confirm the modification

Postconditions: None

3.1.3.7 See the Favorite List

Actors: Users

Goal: Users want to see their restaurant favorite list.

Preconditions: Users are logged in

Main Success Scenario:

1. Users choose to see their favorite list.
2. The system shows a list of their favorite restaurants to the users.

3. Users see their favorite list.

Extensions (a)

3a. Users choose to remove the restaurant from their favorite list.

4a. The restaurant is removed from their favorites list.

Postconditions: None

3.1.3.8 Logout

Name: Users Logout

Actors: Users

Goal: Users want to logout from the system.

Preconditions: Users are logged in.

Main Success Scenario:

1. User choose to logout

2. User is logged out

Postconditions: None

3.1.4 Functional Breakdown Structure

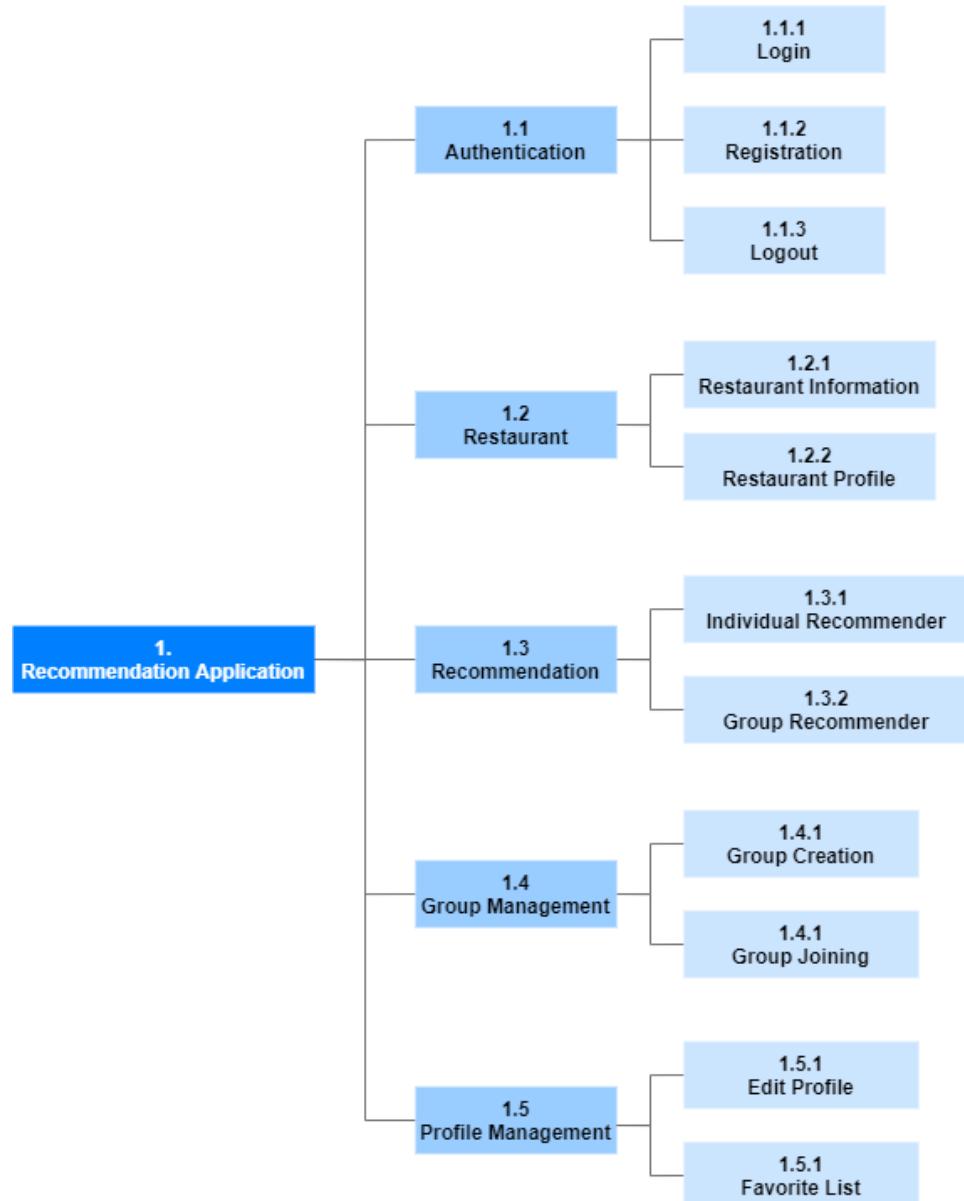


Figure 3.2 Functional Breakdown Structure of the System

Our system consists of 5 functional components which are:

1. Authentication, it handles user registration, login, and logout.
2. Restaurant, it serves the restaurant information and their profile which include restaurant categories, price and distance.
3. Recommendation, it generates the restaurant recommendation which is separated into individual and group recommendation.
4. Group Management, manages the group used in group recommendation.
5. Profile Management, manages the user profile and their favorite list.

3.2 System Architecture

3.2.1 System Overview

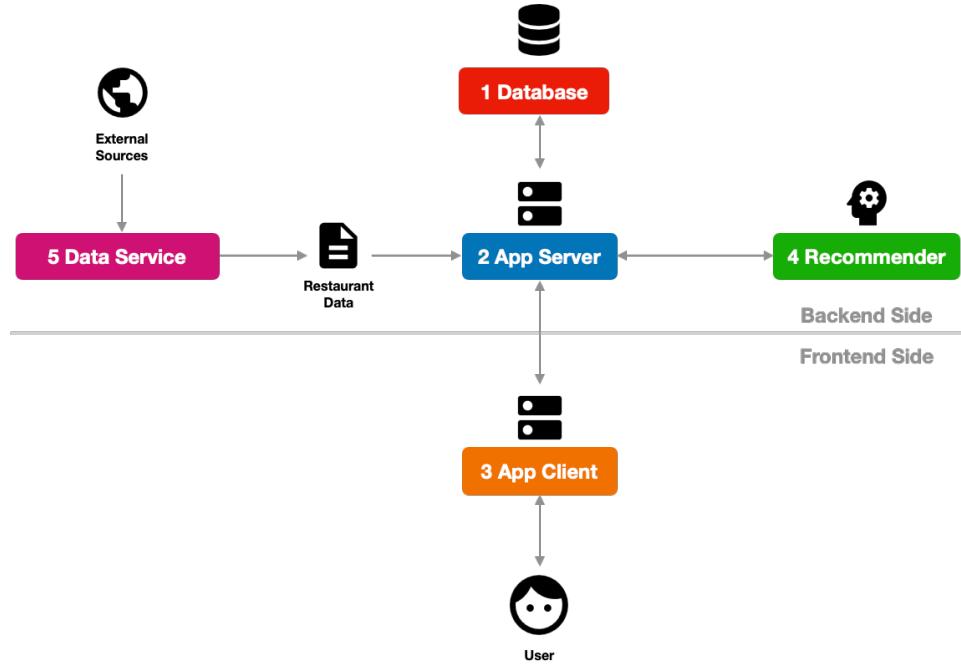


Figure 3.3 System Overview

In this section, we will give an overview of our system about their responsibility and we will discuss each component in detail in the next section.

From Figure 3.3, we can see the relationship between each component. Our system consists of mainly 5 components which are Database, App Server, App Client, Recommender, and Data Service.

- Database: store all of the data of our system.
- App Server: handle all server-side processes including restaurant data management, group management, recommendation request handling, and authentication system. This component acts as a center of the system.
- App Client: serve the user interface to users via a web application.
- Recommender: handle the restaurant's suggestion. This component will receive the recommendation request from App Server and return the suggestion result back to App Server.
- Data Service: collect restaurant data from external sources, preprocess the data, and feed the data into App Server.

We separate these components into 2 sides which are the backend side and the frontend side. The backend side includes Database, App Server, Recommender, and Data Service. The frontend side is only App Client.

3.2.2 Restaurant Recommender System Architect

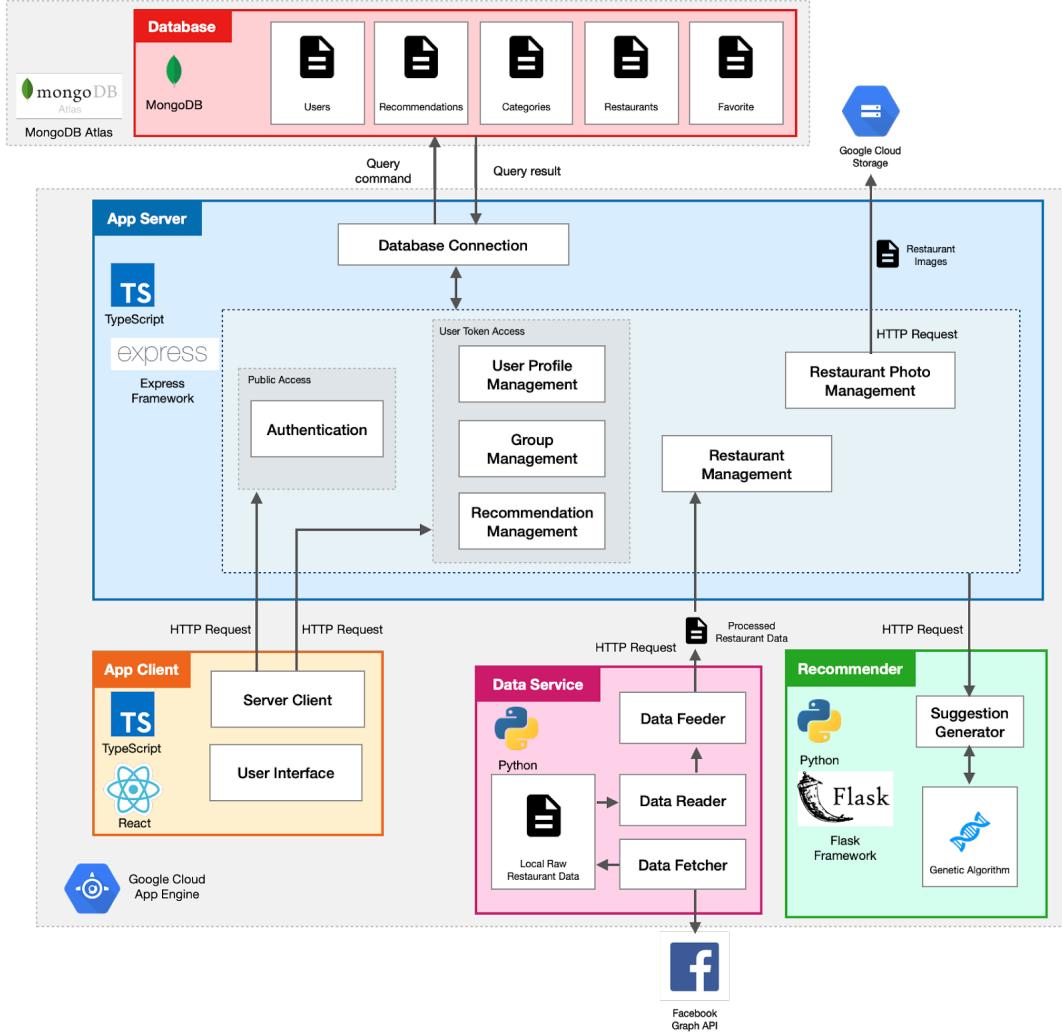


Figure 3.4 System Architecture

Figure 3.4 shows the detailed architecture of our system. As we mentioned before, our system consists of 5 components. Each of these components communicates via HTTP request and response. All of the components except Database are deployed using Google Cloud App Engine and Database is deployed via MongoDB Atlas. There are 2 external services that our system needs to be interacted with which are Google Cloud Storage for storing restaurant's images and Facebook Graph API for querying primary restaurant information. We will walk through each component in detail in the following section.

3.2.2.1 App Server Architecture

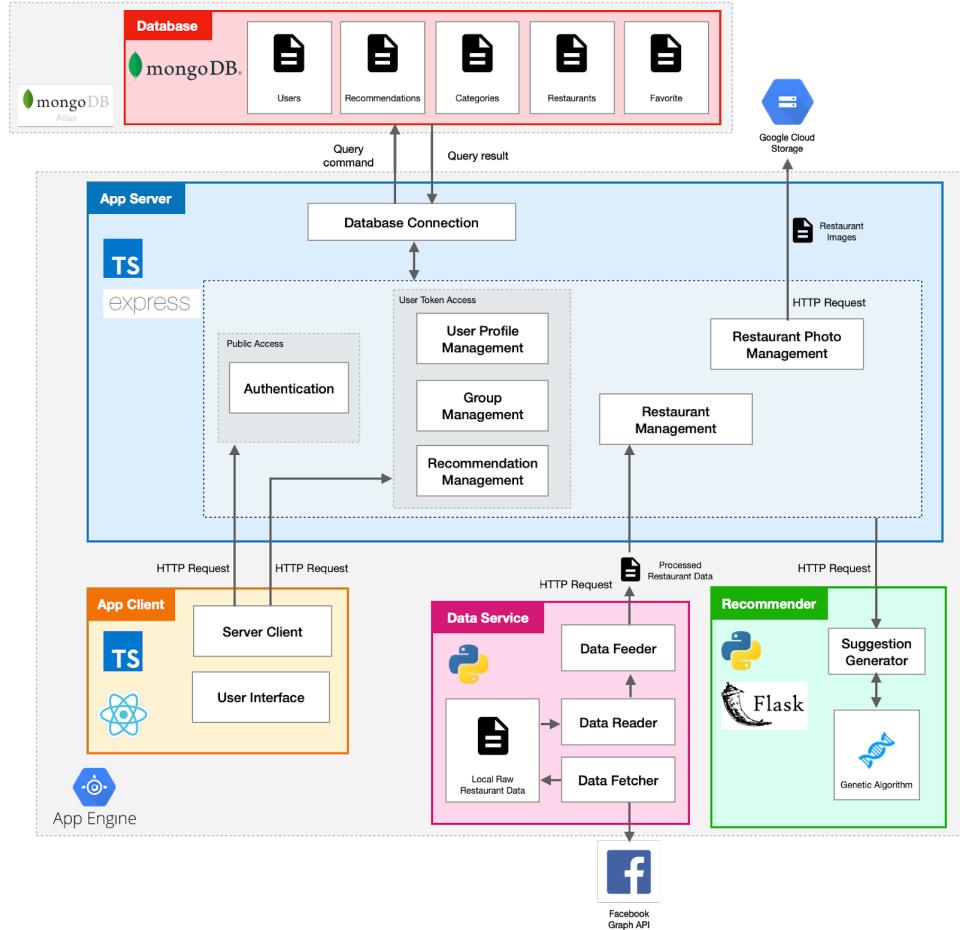


Figure 3.5 App Server Architecture

App Server is responsible for handling all server-side processes. The processes include authentication, user management, group management, recommendation management, restaurant management, and model management. This component acts as a center of our system that lets all other components communicate with them.

For database communication, this is the only component that allows access to the database, so any components needed to access any data must communicate through App Server. The database connection module acts as an interface to the database for App Server.

For an App Client communication, App Client needs to send a request to App Server in order to complete a task. The requests include authentication requests, create and update a recommendation request, get, create, and update group requests and get, create, and update user information requests.

For a Data Service communication, Data Service generates processed restaurant data and it sends those data to App Server to create a restaurant and store them in the database.

For a Recommender communication, App Server needs to send requests to Recommender for restaurant suggestion generation and model training. Recommender also needs to send a request to App Server to get restaurant information and user information.

There is also one external system that App Server needs to communicate with which is Google Cloud Storage. App Server will send a request to store the restaurant's images and get the existing image from that service.

App Server is implemented using TypeScript and Express framework which is a web application framework and deployed on Google Cloud App Engine service.

3.2.2.2 App Client Architecture

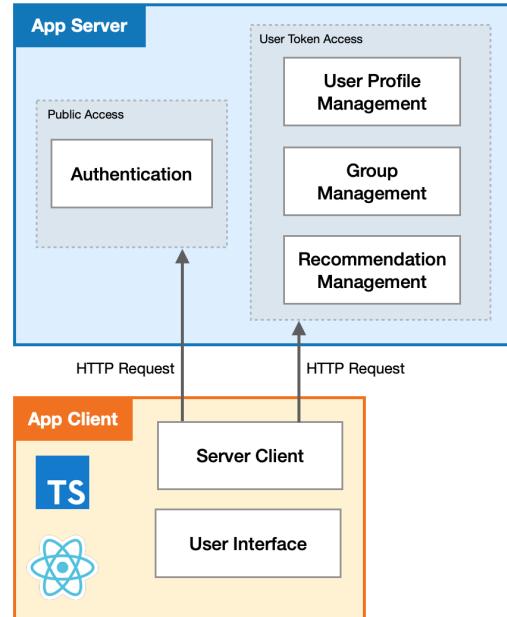


Figure 3.6 App Client Architecture

App Client is responsible for serving the user interface to users and handling user interactions. There are 2 modules in this component which are User Interface that serves as an interface to users and Server Client that handles user's tasks and communicates with App Server. App Client is implemented using TypeScript and React framework and deployed on Google Cloud App Engine service.

3.2.2.3 Recommender Architecture

Recommender is responsible for restaurant suggestion generation. There are 2 modules in this component which are Suggestion Generator and Model Manager.

Suggestion Generator will receive a recommendation request from App Server, generate one set of restaurants using a local model, and send that response back to App Server. In this process, Recommender also needs to get some information about restaurants and users from App Server for the generation.

Model Manager will manage the local model. It is waiting for a training request from App Server to train a new local model. It also sends the training information request back to App Server after the training process is finished.

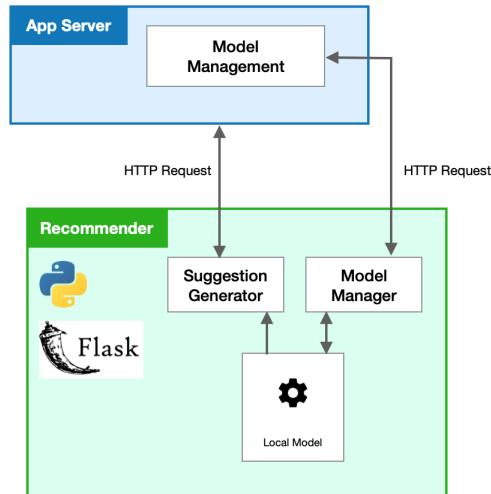


Figure 3.7 Recommender Architecture

Recommender implemented using Python which can easily communicate with the model because many libraries are supporting the machine learning process. It also uses the Flask framework to make them listen for requests from App Server. It is deployed on the Google Cloud App Engine service.

3.2.2.4 Data Service Architecture

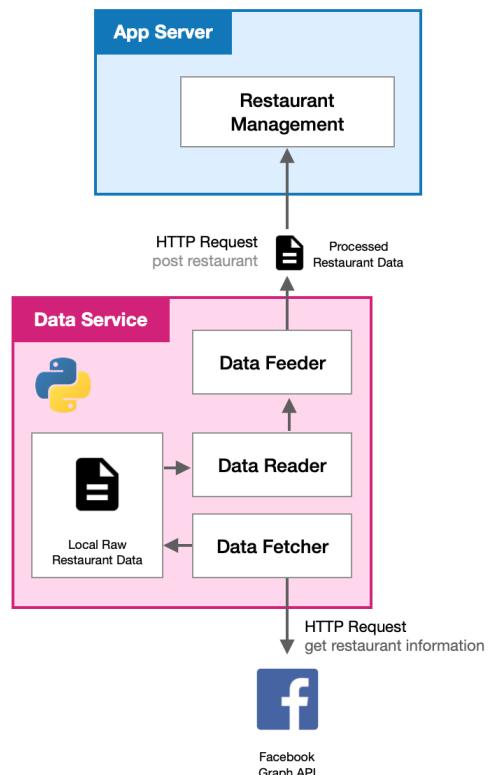


Figure 3.8 Data Service Architecture

Data Service is responsible for collecting restaurant information from an external source. This component has 3 modules which are Data Fetcher which collect the restaurant information from Facebook Graph API and save it to the local raw files, Data Reader which will read the files and process that data into the correct format and Data Feeder will send the processed data to App Server to create restaurants in the system. Currently, it is designed to collect the restaurant data manually, so if we want to update or collect more restaurants, we need to trigger the task in this service again. Data Service is implemented as a simple Python script without any web framework because no components need to send a request to Data Service.

3.2.2.5 Database Architecture

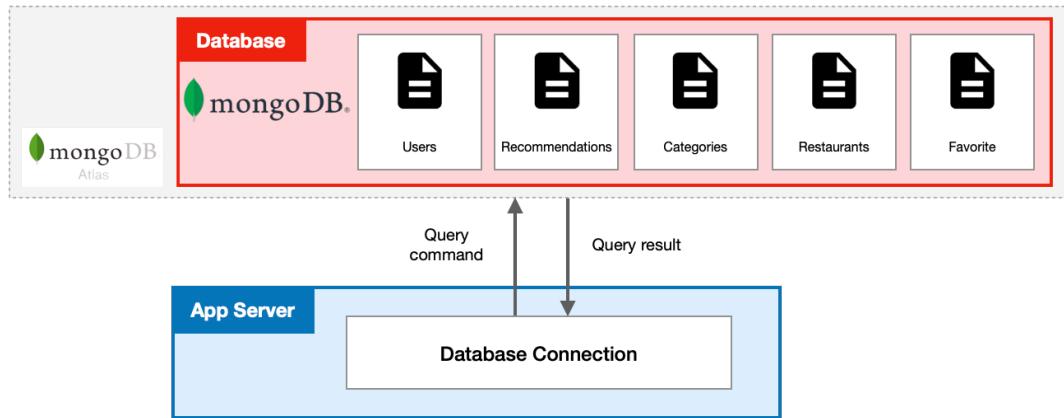


Figure 3.9 App Server Architecture

Database is responsible for storing all data in our system. We choose to have NoSQL for our database because the schema of the data can change quickly, having a flexible schema can make our system adapt to the change in data format more easily. We will discuss the database structure in section 3.5 Database Structure. To access the data in the database, App Server will send a query command via the database connection module in App Server. The database is implemented using MongoDB and deployed using their official cloud service, MongoDB Atlas.

3.3 User Journey

3.3.1 Activity Diagrams

3.3.1.1 Registration

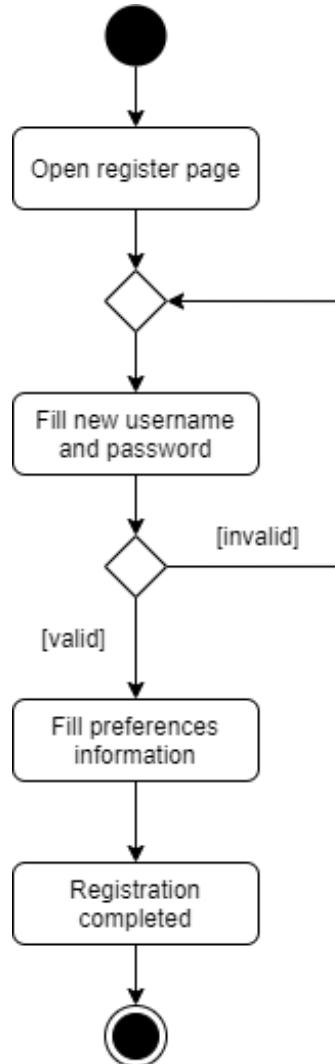


Figure 3.10 Registration Activity Diagram

When users open a registration page, they will be asked to input their new username and password. If their new username and password are valid they need to fill in the food preferences information. Then the registration will be completed.

3.3.1.2 Individual Restaurant Recommendation

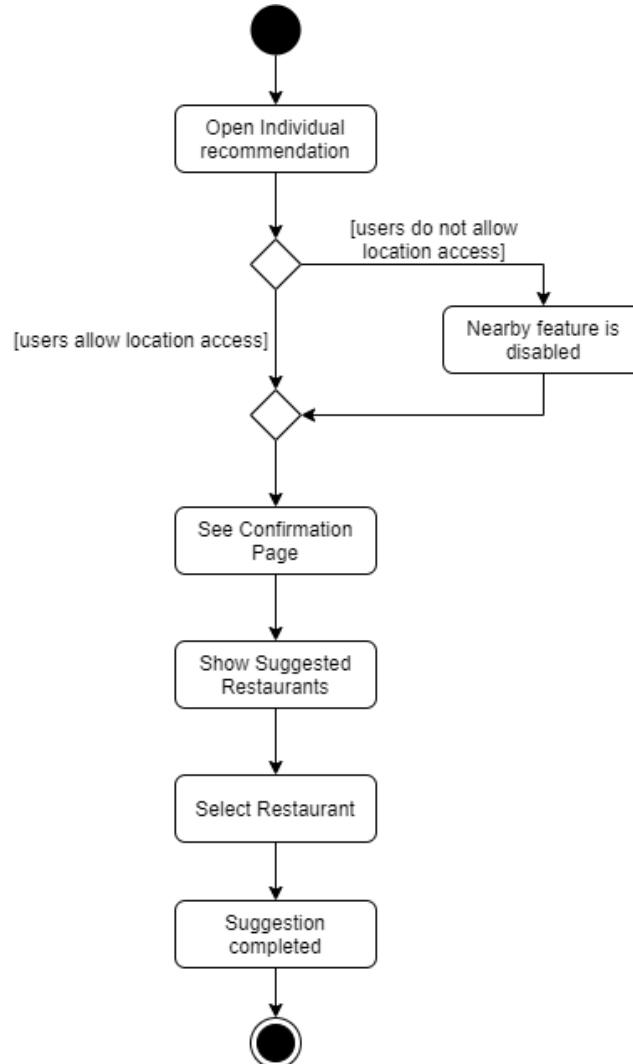


Figure 3.11 Individual Restaurant Recommendation Activity Diagram

When users choose to start the individual recommendation, users need to allow their current location access in order to use nearby suggestion features. And before entering the recommendation, they will see the confirmation page which they can configure the setting and location. After that, they will see the suggested restaurant and can optionally save any restaurant into their favorite list. If they are satisfied with the suggestion and select the final restaurant, the recommendation will be completed. Otherwise, users can choose to see more suggestions.

3.3.1.3 Registration

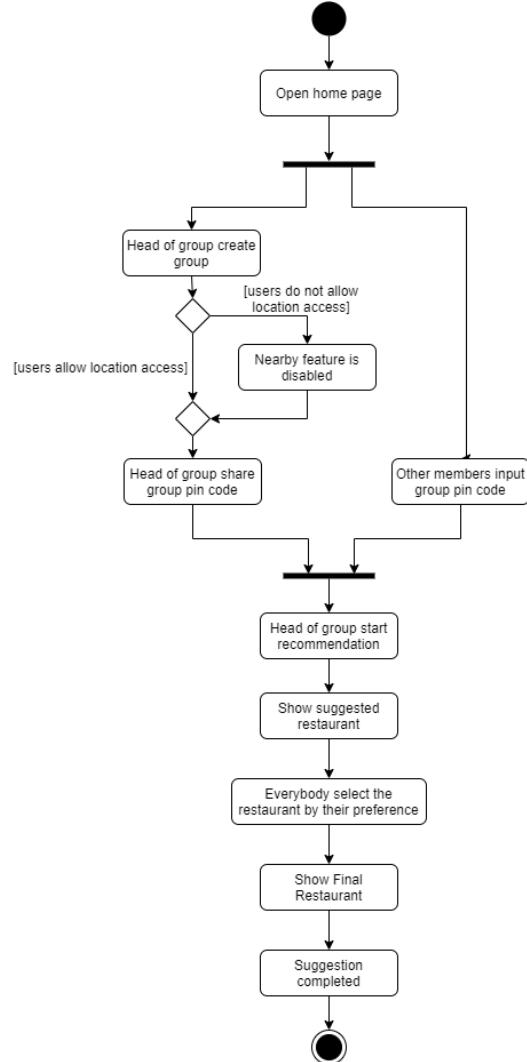


Figure 3.12 Group Restaurant Recommendation Activity Diagram

When a group of users wants to start a group recommendation, they firstly need to create a group and let other members join the group. This requires only one member to create a group and that person will be a head of the party which they can change the location and main configuration. Other members can use the group pin code shared by the head or other members that are already in the group. After the head chooses to start the recommendation, the same set of restaurants will be shown to all members. Everybody then needs to rank the restaurants that they want to go to by their preference. After everybody is done selecting their restaurants, the final suggested restaurant will be shown to the group and the recommendation is completed.

3.3.2 Sequence Diagrams

3.3.2.1 Registration

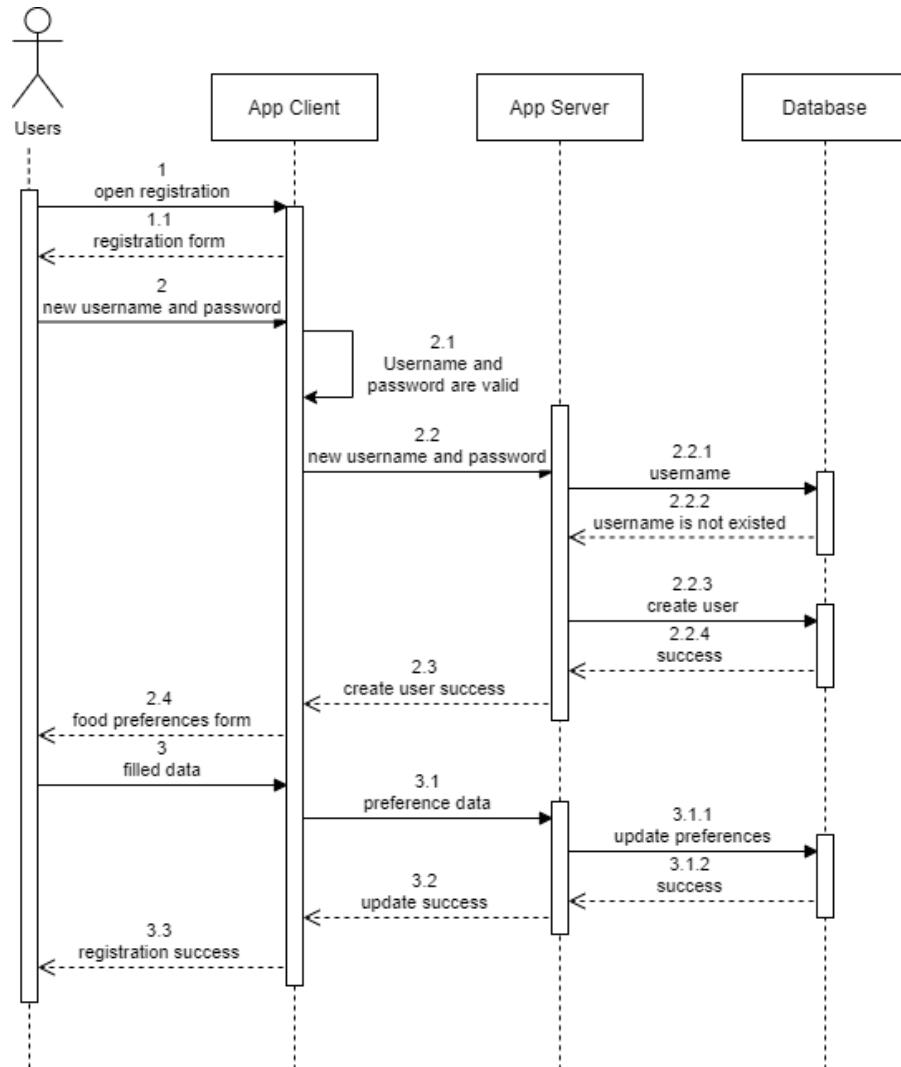


Figure 3.13 Registration Sequence Diagram

Scenario: Users register for a new account.

1. Users open the registration page and App Client shows the registration form to users.
2. Once users fill in their new username and password, App Client will validate the username format (2.1). If they are valid, App Client will send those fields to the app server (2.2). Next, App Server will query the username from the database (2.2.1). If the username does not exist in the database, the new user will be created (2.2.4) and sent back to App Client (2.3). The App Client will send a food preference setting form to the users (2.4).
 - 3.1.1 update preferences
 - 3.1.2 success
3. Once users fill the food preferences setting form, the user profile in the database will be updated (3.1.2) and the registration will be completed (3.3)

3.3.2.2 Individual Restaurant Recommendation

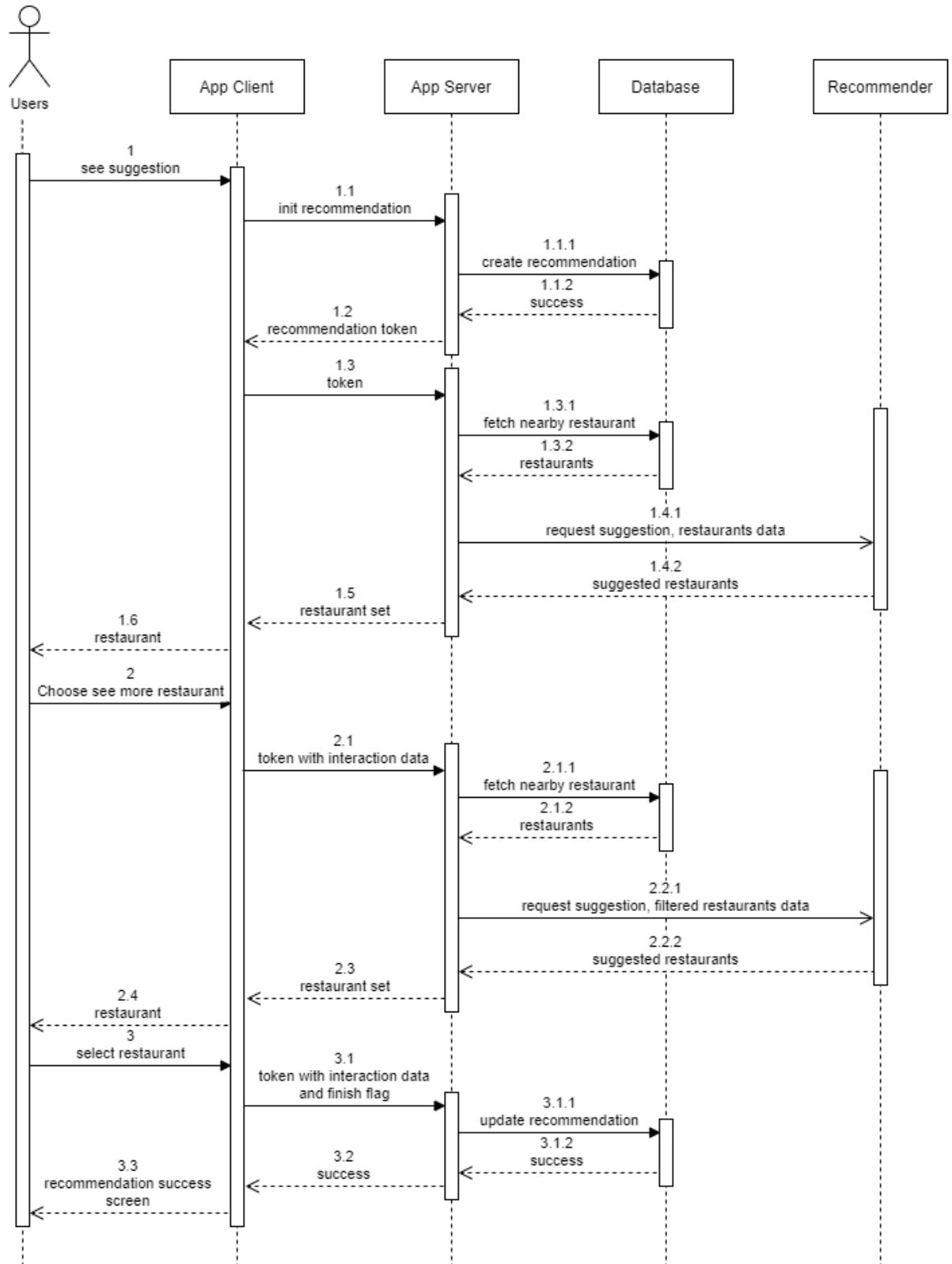


Figure 3.14 Individual Restaurant Recommendation Sequence Diagram

Scenario: Users want to see the restaurant recommendation for individuals and they want to see more suggestions than the first set of the suggested restaurants.

1. When users want to see a recommendation, App Client will send an initial recommendation request to the App Server (1.1), App Server will create a new recommendation data (1.1.1 and 1.1.2) and send the recommendation back to App Client (1.2). Next, App Client will send the token to App Server again (1.3). Once App Server receives the token, it will fetch nearby restaurants (1.3.1 and 1.3.2) and send a recommendation request with the restaurants to Recommender (1.4.1). Recommender will generate a set of suggested restaurants and send them back to App Server and App Client (1.4.2 and 1.5). Lastly, App Client will show a restaurant in the restaurant set to users (1.6).
2. Users will see the suggested restaurants. In this scenario, users choose to see more suggested restaurants (2). After that App Client will send a recommendation token with current restaurants interaction data to filter out the suggested restaurants in the first set (2.1). The next process is generating a restaurant suggestion which is the same process as before, but this time, App Server will send only the filtered restaurants to Recommender (2.2.1). After the suggestion is generated, it will be sent back to App Server, App Client and users (2.2.2, 2.3, and 2.4).
3. When users select the restaurant from the set (3) which means the recommendation is finished, App Client will send the recommendation token with a finish flag to App Server (3.1). App Server will send an update request to Database (3.1.1) and after its success, the recommendation success screen will be shown to users (3.3).

3.3.2.3 Create Group

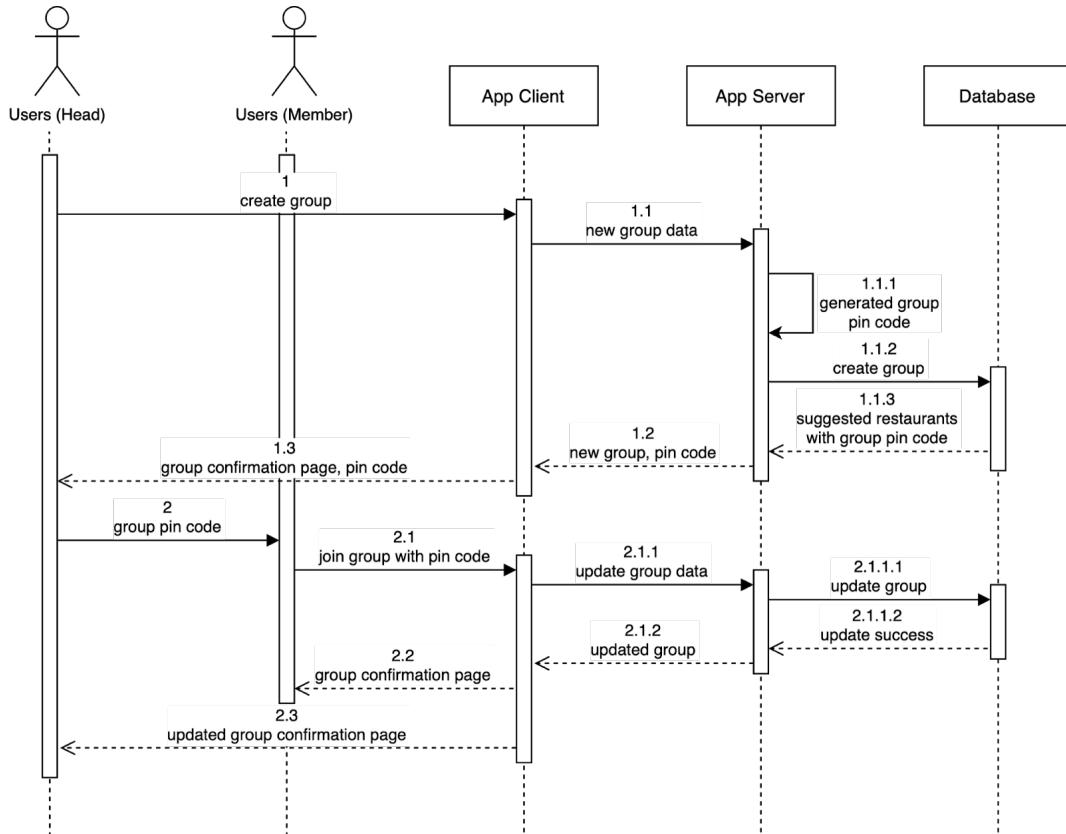


Figure 3.15 Create Group Sequence Diagram

Scenario: A group of 2 people wants to create a new eating group by the first user creating a group and the other one joining the group.

1. The first member creates a group (1). App Client sends data to App Server (1.1). App Server generates a group pin code (1.1.1) and creates a new record in the database (1.1.2). Finally, the member will see a group confirmation page and a group pin code (1.3).
2. After the member has the group pin code, they can share it to other members (2). When other members join a group with a pin code (2.1), App Server will update the record in the database (2.1.1) and other members will see the same group confirmation page as the person who created a group (2.2). The group information of the first member will also be updated (2.3).

3.3.2.4 Group Restaurant Recommendation

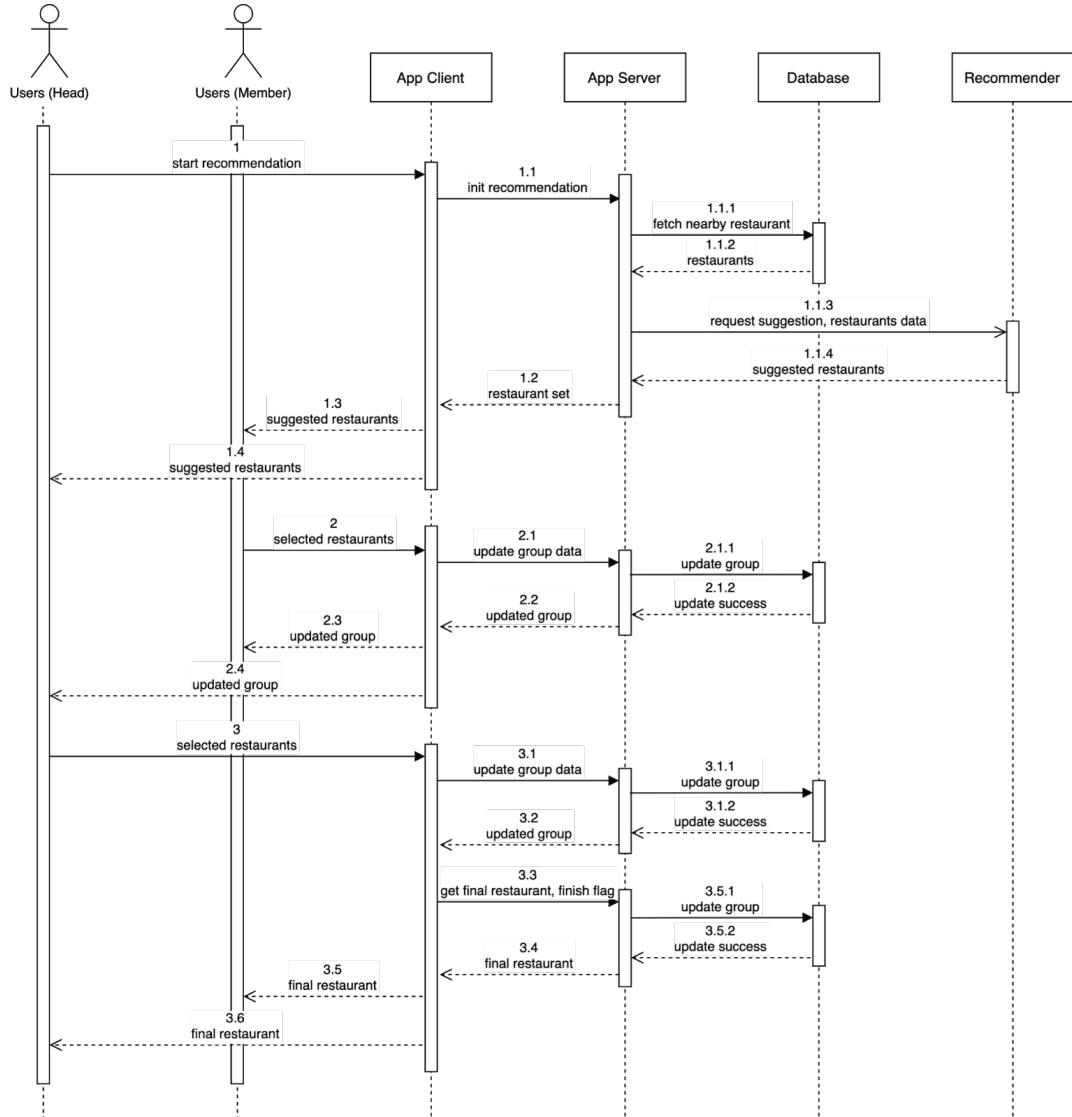


Figure 3.16 Create Group Sequence Diagram

Scenario: A group of 2 people wants to have a restaurant recommendation for their group and they already have a group created.

1. When a head of the group which is the first member who created the group chooses to start a recommendation (1), App Client will request a new group recommendation to App Server (1.1). After that App Server and Recommender will generate a suggestion which has the same process as the individual recommendation (1.1.1 to 1.1.4). Note that Recommender can handle multiple users in the recommendation. Finally, the same suggested restaurant set will be shown to all members (1.3 and 1.4).
2. The first member finishes selecting their restaurant by their preference (2). App Server will update a group data (2.1) and both of the members will receive the update group data (2.3 and 2.4). However, the recommendation is not completed yet, it needs to wait for the other member to finish selecting the restaurants.
3. The second member finishes selecting the restaurants (3). App Server will update a group data again (3.1). In this time, App Client detects that every member has finished selecting the restaurant, so it will send a group finalization request with a finish flag (3.3). App Server will finalize the final restaurant to the group (3.3.1) and update the group data again (3.3.2). Finally, the final restaurant will be shown to every member (3.5 and 3.6) and the recommendation will be completed.

3.4 User Interface Design

This section presents the user interface design of the restaurant recommendation system which designs in the Figma application.

The user interface of our web application includes login, registration, individual restaurant recommendation, group restaurant recommendation, and users' profile as follows.

3.4.1 Login Page

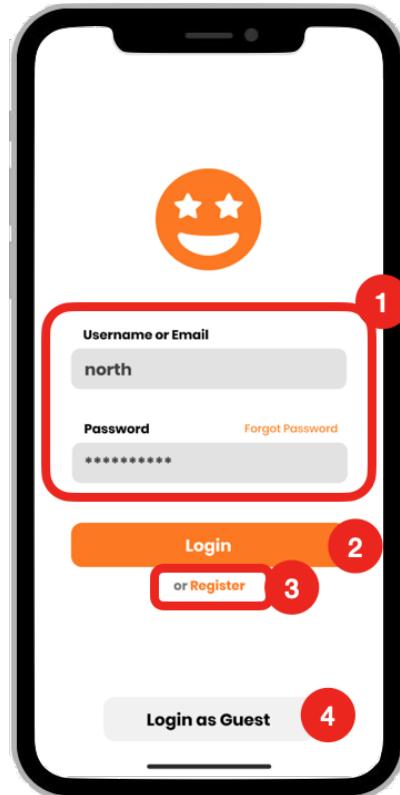


Figure 3.17 Start Page User Interface Design

1. Username and password input fields
2. Login button to confirm the input fields
3. Register button, it links to the registration page
4. Login as Guest for users who do not want an account but want to start using our system right away.

When users open the application, the login page will appear first if they did not login yet. They need to input their username and password in the fields (1) then They can tap the “Login” button (2) to log in. After login, they will see the home page (Figure 3.19). If they do not have an account, they can tap on the “Register” button (3) and they will see the registration page (Figure 3.17). However, they can choose to login without any account by tapping on the “Login as Guest” button (4) and the system will directly bring them to the home page (Figure 3.19).

3.4.2 Registration Page

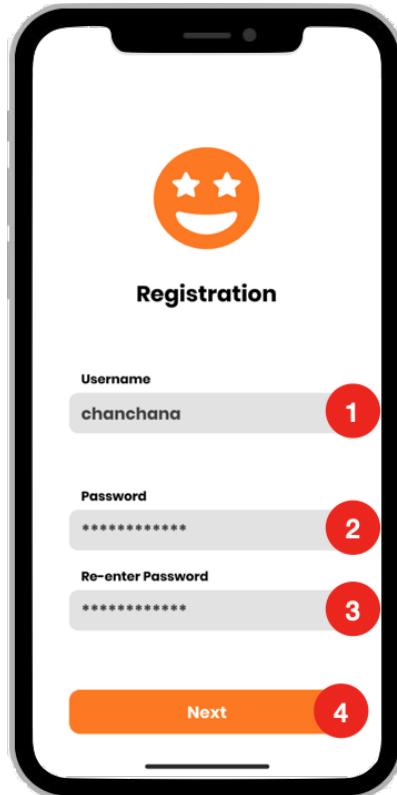


Figure 3.18 Registration Page User Interface Design

1. New username field
2. New password field
3. Re-enter the password field, it needs to be matched with the password field
4. Next button, links to a general information setup page

If users choose to register, the registration page will appear and ask users to input a new username, password, and re-enter the password (1), (2), (3) respectively. If the input username is invalid or already taken, the system will alert users and let them choose a new username. If the password field and re-enter password field are mismatched, the system will also alert and tell users to input their password and re-enter password fields again. After all of the information in the fields is valid and users tap on the “Next” button (4), they will proceed to the next step which is the profile setting step stated in 3.19.

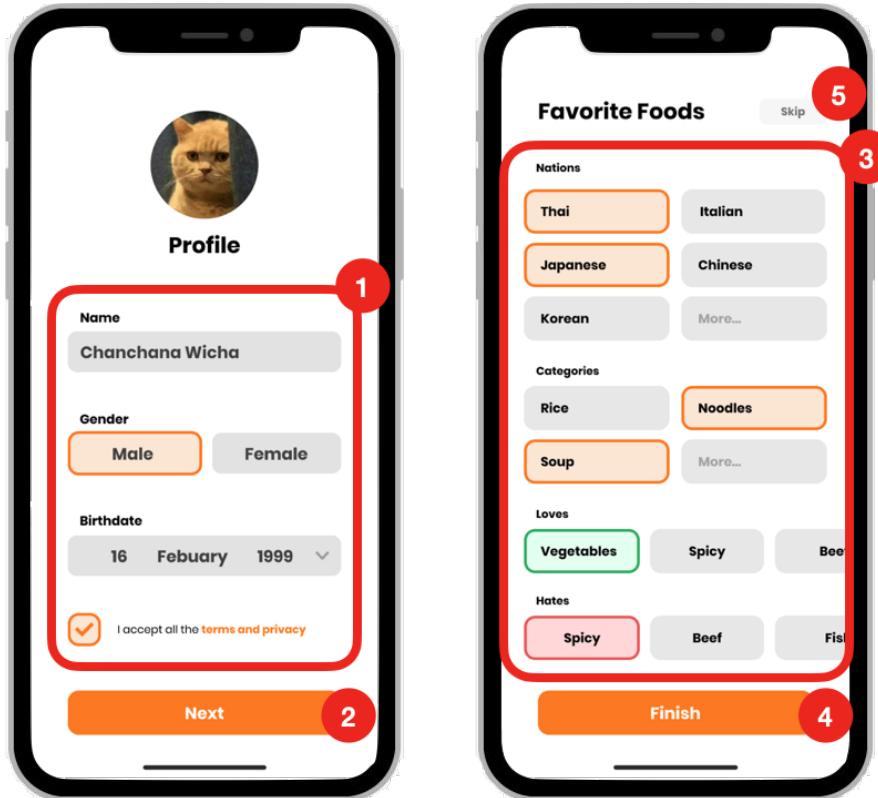


Figure 3.19 Profile Setup Page User Interface Design

1. General information form
2. Next button, will proceed users to the food preference setup step
3. Food preference form, let users choose their food preference in each category.
4. Finish button, to complete the form and the registration.
5. Skip button, lets users skip the food preference setup process.

After users input the valid username and password, the next step is user profile setup. This process consists of 2 steps which are general information setup and food preferences information setup. The first one on the left of 3.19 is a general information setup step. Users are required to fill in all the information in the form and accept the terms and privacy (1) then they can proceed to the next step by tapping the “Next” button (2). For the second step, preferences information set up, they will see many available categories of food. They can choose their food preferences in the form (3) and tap on “Finish” to complete the registration. However, if they do not want to set their preferences yet, they can tap on the “Skip” button (5) to skip the preferences setup process. After the registration is completed, they will see the home page (Figure 3.19) and start using our system.

3.4.3 Individual Restaurant Recommendation Page



Figure 3.20 Home Page User Interface Design

1. Restaurant card, it shows the general information about a restaurant.
2. “i” button, when tapped, more information will be shown.
- 3.
4. Like button, tapped once users are satisfied with the restaurant
5. Dislike button, if users are not satisfied with the restaurant, they can tap this button to see the next suggestion.
6. Profile button, links to the profile page.

After users logged in or registered, they will see the home page. By entering the home page, the system will automatically start recommending some restaurants to users. Users will see the information of each suggested restaurant (1). They can also see some more information about each restaurant by tapping the “i” icon (2). They can filter the suggested restaurants by tapping the “Filters” button (3) and the filter window will appear (Figure 3.20). If they are satisfied with the suggested restaurant, they can tap on the “Yummy” button (4) or swipe the restaurant card to the right, and then the success page will appear (Figure 3.21). Otherwise, if they are not satisfied with that restaurant, they can tap on the “Nah” button (5) or swipe the restaurant card to the left, and then the system will show the next suggested restaurant to users and let them consider it again. Users can see their profile by tapping the profile picture button (6) and the profile management page will appear (Figure 3.25).

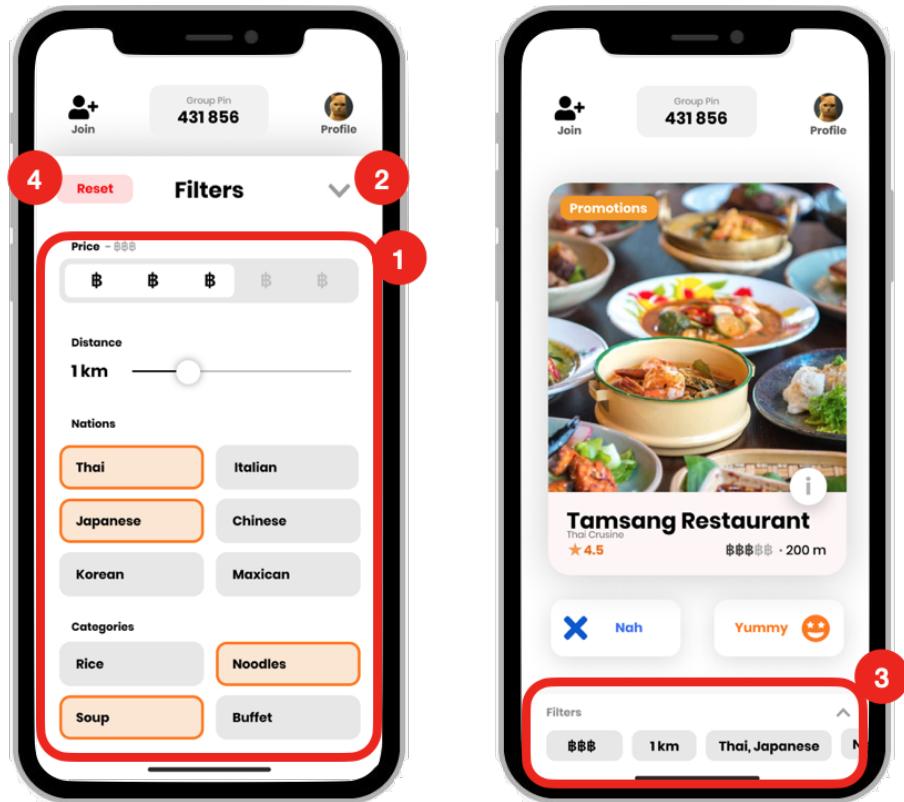


Figure 3.21 Filter Window User Interface Design

1. Filtering options.
2. Arrow button, will apply current filtering and close the filter window.
3. Filter button with applied filter tags, users can open the filter window again by tapping this button.
4. Reset button, will remove all the applied filters.

After users tap on the “Filter” button on the home page (number 3 on 3.20), the filter window will appear. Users can filter restaurants by choosing the available options that appear on the filter window (1). After finishing choosing the filtering option, they can tap on the arrow icon (2) to apply the filter and the filter window will be minimized. They can see the applied filters at the bottom of the screen (3). They can change their filtering by tapping on the bottom bar (3) to bring the filter window up again. They can tap on the “Reset” button (4) to remove all of the applied filters. After applying any filters, the system will suggest the restaurant that matches its filtering condition.



Figure 3.22 Recommendation Success Page User Interface Design

1. Swipe right for the satisfying restaurant.
2. Like button which works the same as the swipe right.
3. Restaurant card showing general information about the liked restaurant.
4. Recommendation rating, let users rate satisfaction with their current recommendation.

On the home page which users see the suggested restaurant, if users are satisfied with the suggested restaurant, they can swipe right (1) or tap on the “Yummy” button (2). After that, the recommendation success page will appear (the right picture in 3.22). On the success page, users can see the restaurant information in the center of the screen (3). They can give a rating score for the recommendation from 1 star to 5 stars (4). After they are done seeing the information, they can tap on the “Done” button (5) to close the success page and they will see the home page (Figure 3.19) again.

3.4.4 Group Restaurant Recommendation Page

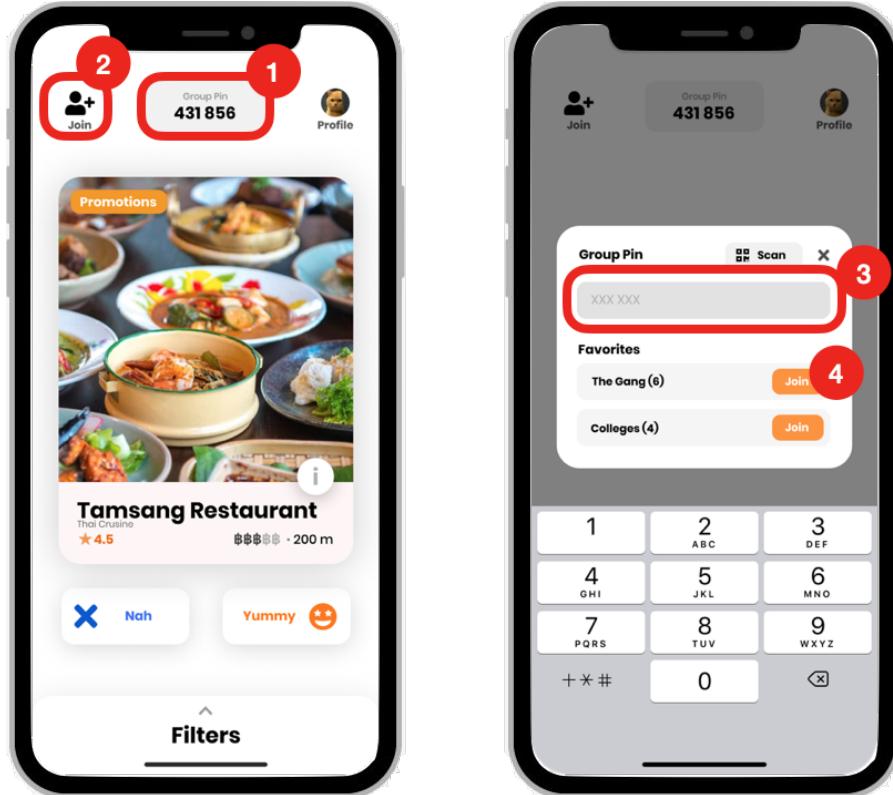


Figure 3.23 Group Joining User Interface Design

1. Group pin code which other members used for joining the group.
2. Join button, it will bring up the joining dialog
3. Group pin code input field, let users input the group pin code.
4. Join button which lets users join their existing group without input the group pin code again.

In the following section, we will walk through the group recommendation process. On the home page (the left picture in 3.23), it will show the group pin on the top of the screen (1). To start the group recommendation, the user has to share this pin code with other members in the group and this will make the pin code owner be the head of the group. Other members can join the group by tapping on the “Join” button (2) and the joining dialog will appear (the right picture in 3.23). In the joining dialog, other members can input the pin that appeared on the group head’s device (1). Moreover, users can join the previously joined group, which will be shown as favorites. They can directly join their favorite group by tapping the “Join” button (4) in the group they want to join without input the pin code again. After someone joins the group the group recommendation will be initialized and all members that joined the group will see the group confirmation page next (Figure 3.23).

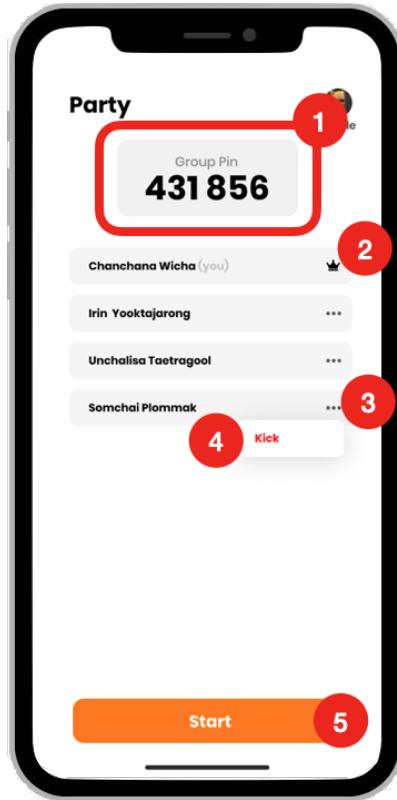


Figure 3.24 Group Confirmation Page User Interface Design

1. Group pin code which other members used for joining the group.
2. Crown icon indicates who is the head of this group
3. 3 dots icon for more action on each member in the group.
4. Kick button, it let the head remove some members from the group.

After someone joins the group, all joined members will see this group confirmation page. On the top, there is a group pin code that the members in the group can share with other people if they want to join this eating group. The head of the group will have a crown icon (2) and they can manage the members in the group. If the head wants to remove some of the members, they can do it by tapping the 3 dots icon (3) followed by tapping the “Kick” button, after that the kicked member will no longer be in the party. When all members are joined and ready, the head will tap on the “Start” button to begin the group recommendation.



Figure 3.25 Group Recommendation Page User Interface Design

1. Like or dislike area for choosing the restaurant.
2. Progress bar indicates the progress of the group decision.
3. Members button, will bring up the members dialog which shows all members.
4. Filter button for setting the filters for restaurants.

After the group has been created and confirmed, the group recommendation will begin. All members will see the same suggested set of restaurants. Each member will choose whether they like or not but tapping “Yummy” and “Nah” or swiping (1) is the same as in individual recommendation. The group recommendation will be finished after all members like the same restaurant. The top bar (2) indicates the highest number of members that agree on the same restaurant, when it is full, it means everybody agrees on the same restaurant and the recommendation is finished. Users can see the members by tapping the “Member” button (3). Users still can set the filter by tapping the “Filter” button (4) but this time the filters will be applied to every member in the group.

3.4.5 Profile Page

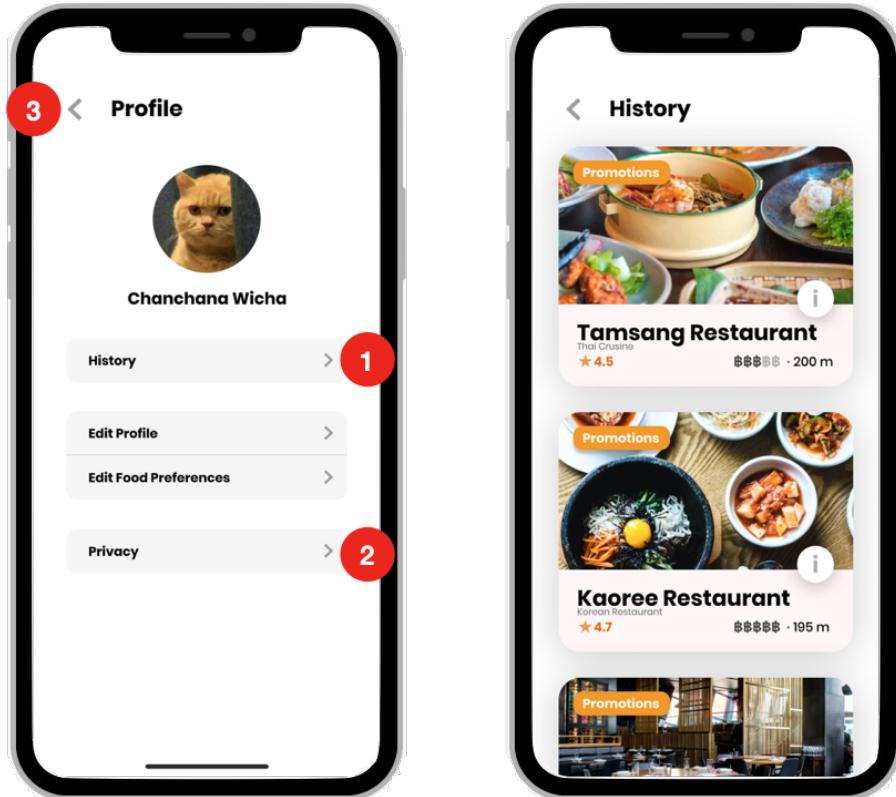


Figure 3.26 Profile Management Page User Interface Design

1. History button, it links to the history page (the right picture in Figure 3.25)
2. Privacy button, it let users change their password and their privacy setting.
3. Back button which brings users back to their home page.

After users choose to view their profile (number 6 in 3.20), they will see the left picture in 3.26. They can choose to view their selected restaurant history by tapping the “History” button (1) and a list of selected restaurants will be shown (the right picture of 3.26). They can also choose to edit their profile such as their name, gender, or birthdate or choose to edit their food preferences. They can reset their password and set the privacy setting under the “Privacy” section (2). They can go back to the main screen by tapping the arrow icon (3).

3.5 Database Structure

In this section, we explain our database structure in the Database component. Our database stores all information for our system as a NoSQL database. Then, we discuss the database collection’s name and description, the schema of each collection, and database indexing.

3.5.1 Collections

Table 3.1 Database Collection and Description

Collection Name	Description
restaurants	Restaurant information including name, address, photo URL, etc., restaurant profile.
users	User information including authentication information, general information such as name, gender, age, and food preference information.
recommendations	Information for each recommendation session. This collection includes all of the interaction histories from the user in each recommendation session.
categories	Information about restaurant categories. We keep the categories separated with restaurant collection to reduce the redundancy.
favorites	Keep the information about the favorite list of restaurants for each user.

Table 3.1 shows us all collections in our NoSQL database, their collection's name, and their description. Our database contains mainly 5 collections including restaurants, users, recommendations, categories, and favorites.

3.5.2 Schema

Although NoSQL does not have a fixed database schema, by using Mongoose as a MongoDB database connection framework, it provides some primary schema for data creating and reading facilitation. Having Mongoose's schema can help us manage a uniform data format with easy schema migration. However, inserting data with a different format from Mongoose's schema does not break the process and no part of the data will be discarded. In this section, we will show some mandatory fields for each collection.

Table 3.2 Restaurant Schema

Key Name	Type	Description
name	String	Name of the restaurant
profile.categories	Array of ObjectId	Restaurant categories such as Thai, Korean, Fast Food. This field references category collection using their category id.
profile.price_range	Number	The price range of the restaurant, from 1-4.
profile.rating	Number	Facebook users' rating of the restaurant. From 1 to 5 stars.
profile.likes	Number	The number of users that liked the restaurant's Facebook page.
address	String	One line address string of the restaurant.
location.coordinates	Array of Numbers	Coordinate of the restaurant in form of [longitude, latitude]
link	String	URL link to their Facebook page.

Table 3.3 User Schema

Key Name	Type	Description
authentication	Object	Authentication token object for user login.
username	String	Username of the user.
password	String	Encrypted password of the user.
profile.gender	String	Gender of the user.
profile.birthdate	Date	Birthdate of the user.
profile.preference	Object	Food preferences of users.

Table 3.4 Recommendation Schema

Key Name	Type	Description
users	Array of ObjectId	Users that request the recommendation. This field references user collection using their user id.
histories	Array of Object	User interaction history in each recommendation session.
histories.restaurant	ObjectId	Interacted restaurant. This field references restaurant collection using their restaurant id.
histories.is_love	Boolean	Whether users select go or not go to the interacted restaurant.
histories.timestamp	Date	The timestamp of each interaction history.
location.coordinate	Array of Numbers	The location that this recommendation session occurs in the form of [longitude, latitude].

Table 3.5 Category Schema

Key Name	Type	Description
name	String	Name of the category eg. Thai Restaurant, Korean Restaurant, or Fast Food Restaurant.
is_common	Boolean	Whether this category is a common category.

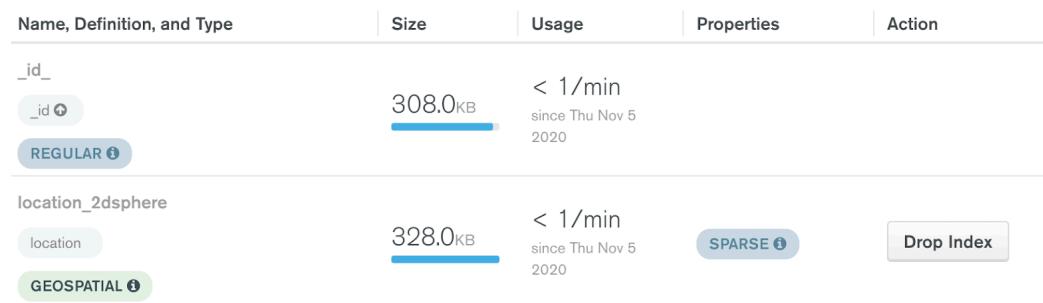
Table 3.6 Favorite Schema

Key Name	Type	Description
user	ObjectId	User of the current record.
restaurants	Array of	List of their favorite restaurants.

Note that we need to keep a category collection separated from restaurant information because some of the collected restaurant data may have a different name but same id or different id with the same name so we want to keep this information correctly and easily maintained later.

3.5.3 Indexing

With database indexing, we can query the data faster by storing additional information about data. Every collection has an indexed id so that we can query the data from their id faster. However, we also set more indexing on 2 collections which are restaurants and users.

**Figure 3.27** Restaurant Collection's Indexes

From [3.27](#), we can see that the restaurant collection has one more index which is the location index. This index is used for query restaurants by a distance of 2 coordinates.

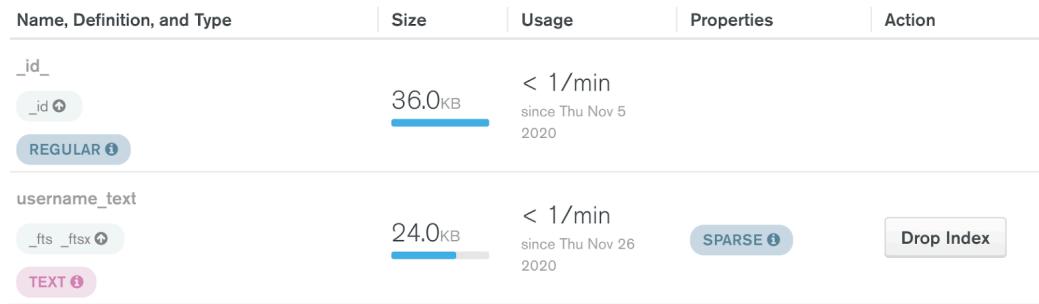


Figure 3.28 User Collection’s Indexes

From 3.28, we can see that the user collection has one more index which is the username index. This index is used for query users by username. It also helps when checking the username’s availability. The username index also applies to other collections that require finding the record by the username which is the favorite collection.

3.6 Restaurant Recommendation Algorithm

3.7 Production Testing Plan

3.7.1 Production Deployment

We will deploy our components that need to listen to requests and send requests to other components which are App Client, App Server, Recommender, and Database. App Client, App Server, and Recommender will be deployed on Google Cloud App Engine since the deployment service is flexible for multiple programming languages and supports auto scalability. Database will be deployed on MongoDB Atlas which is an official cloud deployment service for MongoDB databases. Both Google Cloud App Engine and MongoDB Atlas provide an interactive dashboard for monitoring the performance.

3.7.2 User Evaluation

We will test our application and gather feedback from 20 people. We will collect the data about overall satisfaction, usability, user experience, and improvement suggestions. After the data has been collected, we will analyze and evaluate the application.

3.7.3 Application Evaluation

The performance of main components, App Client, App Server, and Recommender, will be evaluated by monitoring the response latency and doing load tests. The production application performance metrics can also be retrieved by the Google Cloud App Engine dashboard.

3.8 Recommender Systems

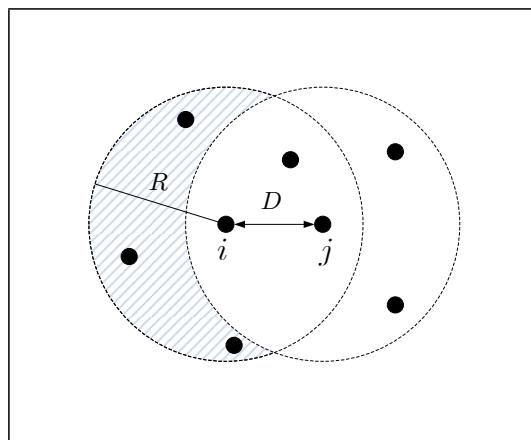
Table 3.7 test table method1

Center	Center	left aligned	Right	Right aligned
Center	Center	left aligned	Right	Right aligned
Center	Center	left aligned	Right	Right aligned
Center	Center	left aligned	Right	Right aligned
Center	Center	left aligned	Right	Right aligned

3.9 Text Processing Algorithms

3.9.1 Algorithm I

You can place the figure and refer to it as Figure 3.29. The figure and table numbering will be run and updated automatically when you add/remove tables/figures from the document.

**Figure 3.29** The network model

3.9.2 Algorithm II

Add more subsections as you want.

3.9.2.1 Algorithm SUB

Add more subsections as you want.

3.10 Development Tools

CHAPTER 4 PROPOSED WORK

Explain the design (how you plan to implement your work) of your project. Adjust the section titles below to suit the types of your work. Detailed physical design like circuits and source codes should be placed in the appendix.

4.1 System Architecture

Table 4.1 test table x1

SYMBOL		UNIT
α	Test variable	m^2
λ	Interarrival rate	jobs/ second
μ	Service rate	jobs/ second

4.2 System Specifications and Requirements

4.3 Hardware Module 1

4.3.1 Component 1

4.3.2 Logical Circuit Diagram

4.4 Hardware Module 2

4.4.1 Component 1

4.4.2 Component 2

4.5 Path Finding Algorithm

4.6 Database Design

4.7 GUI Design

CHAPTER 5 IMPLEMENTATION RESULTS

You can title this chapter as **Preliminary Results** or **Work Progress** for the progress reports. Present implementation or experimental results here and discuss them.

CHAPTER 6 CONCLUSIONS

This chapter is optional for proposal and progress reports but is required for the final report.[\[29\]](#)

6.1 Problems and Solutions

State your problems and how you fixed them.

6.2 Future Works

What could be done in the future to make your projects better.

REFERENCES

1. Margaret Rouse, “What Is Web Application (Web Apps) And Its Benefits,” Available at <https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app>, [Online; accessed 28-November-2020].
2. Expert System Team, “What is Machine Learning?,” Available at <https://expertsystem.com/machine-learning-definition/>, [Online; accessed 26-November-2020].
3. Wikipedia, “Recommender system,” Available at https://en.wikipedia.org/wiki/Recommender_system#Content-based_filtering, [Online; accessed 20-November-2020].
4. Wikipedia, “Collaborative filtering,” Available at https://en.wikipedia.org/wiki/Collaborative_filtering, [Online; accessed 21-November-2020].
5. Sanjam Singh, “Algorithms for Discrete Optimization,” Available at <https://mech.iitm.ac.in/nspch52.pdf>, [Online; accessed 16-March-2021].
6. Wikipedia, “Mathematical optimization,” Available at https://en.wikipedia.org/wiki/Mathematical_optimization, [Online; accessed 16-March-2021].
7. Wikipedia, “Genetic algorithm,” Available at https://en.wikipedia.org/wiki/Genetic_algorithm, [Online; accessed 16-March-2021].
8. Vijini Mallawaarachchi, “Introduction to Genetic Algorithms — Including Example Code,” Available at <https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3>, [Online; accessed 16-March-2021].
9. MDN, “HTML: HyperText Markup Language,” Available at <https://developer.mozilla.org/en-US/docs/Web/HTML>, [Online; accessed 26-November-2020].
10. MDN, “JavaScript,” Available at <https://developer.mozilla.org/en-US/docs/Web/JavaScript>, [Online; accessed 26-November-2020].
11. MDN, “Manipulating documents,” Available at https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Manipulating_documents, [Online; accessed 30-April-2021].
12. Microsoft, “TypeScript,” Available at <https://www.typescriptlang.org/>, [Online; accessed 27-November-2020].
13. OpenJS Foundation, “Introduction to Node.js,” Available at <https://nodejs.dev/learn>, [Online; accessed 27-November-2020].
14. Pete Hunt, “Why did we build React?,” Available at <https://reactjs.org/blog/2013/06/05/why-react.html>, [Online; accessed 27-November-2020].
15. DA14, “TOP 10 ADVANTAGES OF USING REACT.JS,” Available at <https://da-14.com/blog/its-high-time-reactjs-ten-reasons-give-it-try>, [Online; accessed 27-November-2020].
16. Tutorialspoint, “Python Tutorial,” Available at <https://www.tutorialspoint.com/python/index.htm>, [Online; accessed 27-November-2020].
17. Olha Bahaeva, “Top 7 Reasons Why You Need to Learn Python as a Data Scientist,” Available at <https://towardsdatascience.com/top-10-reasons-why-you-need-to-learn-python-as-a-data-scientist-e3d26539ec00>, [Online; accessed 27-November-2020].
18. JupyterLab, “JupyterLab Overview,” Available at https://jupyterlab.readthedocs.io/en/stable/getting_started/overview.html, [Online; accessed 6-December-2020].
19. Compact Creative, “What Is Figma? a 101 Intro,” Available at <https://designshack.net/articles/software/what-is-figma-intro/>, [Online; accessed 6-December-2020].
20. MDN, “An overview of HTTP,” Available at <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>, [Online; accessed 27-November-2020].

21. MDN, “HTTP request methods,” Available at <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>, [Online; accessed 28-November-2020].
22. Wikipedia, “Stateless protocol,” Available at https://en.wikipedia.org/wiki/Stateless_protocol, [Online; accessed 30-April-2021].
23. GoodFirms, “What is a Web Framework?,” Available at <https://www.goodfirms.co/glossary/web-framework/>, [Online; accessed 30-April-2021].
24. IBM StrongLoop, “Express,” Available at <https://expressjs.com/>, [Online; accessed 28-November-2020].
25. The Pallets Projects, “Flask,” Available at <https://palletsprojects.com/p/flask/>, [Online; accessed 28-November-2020].
26. Inc MongoDB, “What is NoSQL?,” Available at <https://www.mongodb.com/nosql-explained>, [Online; accessed 28-November-2020].
27. Guru99, “What is MongoDB? Introduction, Architecture, Features & Example,” Available at <https://www.guru99.com/what-is-mongodb.html>, [Online; accessed 28-November-2020].
28. Mongoose, “mongoose,” Available at <https://mongoosejs.com/>, [Online; accessed 28-November-2020].
29. C.-C. Shen, C. Srisathapornphat, R. Lui, Z. Huang, C. Jaikaeo, and E.L. Lloyd, 2004, “CLTC: A Cluster-Based Topology Control Framework for Ad Hoc Network,” **IEEE Trans. Mobile Computing**, vol. 3, no. 1, pp. 18–32, Jan–Mar 2004.
30. F.O. Isinkaye, Y.O. Folajimi, and B.A. Ojokoh, 2015, “Recommendation systems: Principles, methods and evaluation,” **Egyptian Informatics Journal**, vol. 16, no. 3, 2015.
31. Google, “Google App Engine Documentation,” Available at <https://cloud.google.com/appengine/docs>, [Online; accessed 17-November-2020].
32. Inc MongoDB, “MongoDB Atlas,” Available at <https://docs.atlas.mongodb.com/#service-fullname>, [Online; accessed 18-November-2020].
33. Facebook, “Graph API,” Available at <https://developers.facebook.com/docs/graph-api/>, [Online; accessed 18-November-2020].
34. Sharanya Gopinathan, “Entrée - choose from a world of dishes with a single touch,” Available at <https://techweek.com/entree-choose-dishes-single-touch/>, [Online; accessed 29-August-2020].
35. Apple Store, “Eatsee - See it. Eat it,” Available at <https://apps.apple.com/us/app/id1202110881>, [Online; accessed 30-August-2020].
36. Wikipedia, “Tinder (app),” Available at [https://en.wikipedia.org/wiki/Tinder_\(app\)](https://en.wikipedia.org/wiki/Tinder_(app)), [Online; accessed 30-August-2020].
37. I. Garcia, L. Sebastia, and E. Onaindia, 2011, “On the design of individual and group recommender systems for tourism,” **Expert Systems with Applications**, vol. 38, no. 6, 2011.
38. Chris I, “Build a user-based collaborative filtering recommendation engine for Anime,” Available at <https://towardsdatascience.com/build-a-user-based-collaborative-filtering-recommendation-engine-for-anime-92d35921f304>, [Online; accessed 30-August-2020].
39. Hyun-Tae Kim, Eungyeong Kim, Jong-Hyun Lee, and Chang Wook Ahn, 2010, “A recommender system based on genetic algorithm for music data,” **ICCET 2010 - 2010 International Conference on Computer Engineering and Technology, Proceedings**, vol. 6, 01 2010.
40. L. Bao, “BWorld Robot Control Software,” Available at <http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/>, [Online; accessed 19-July-2008].
41. L. Bao and J.J. Garcia-Luan-Aceves, 2003, “Topology Management in Ad Hoc Networks,” in **Proc. ACM MobiHoc’03**, Maryland, USA, June 2003, pp. 129–140.

42. C. Bettstetter, G. Resta, and P. Santi, 2003, “The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks,” **IEEE Trans. Mobile Computing**, vol. 2, no. 3, pp. 257–269, Jul–Sep 2003.
43. G. Bianchi, 2000, “Performance Analysis of the IEEE 802.11 Distributed Coordination Function,” **IEEE J. Select. Areas Commun.**, vol. 13, no. 3, pp. 535–548, Mar. 2000.
44. P. Gupta and P.R. Kumar, 1998, “Critical Power for Asymptotic Connectivity in Wireless Networks,” **Stochastic Analysis, Control, Optimization and Applications**, pp. 547–566, 1998.
45. H. Honkasalo, K. Pehkonen, M.T. Niemi, and A.T. Leino, 2002, “WCDMA and WLAN for 3G and Beyond,” **IEEE Wireless Commun. Mag.**, pp. 14–18, Apr. 2002.
46. P. Santi, 2005, **Topology Control in Wireless Ad Hoc and Sensor Networks**, Wiley, p.133.
47. I. Norros, 1995, “On the use of Fractional Brownian Motion in the Theory of Connectionless Networks,” **IEEE J. Select. Areas Commun.**, vol. 13, no. 6, pp. 953–962, Aug. 1995.
48. D.Y. Eun and N.B. Shroff, 2001, “A Measurement-Analytic Framework for QoS Estimation Based on the Dominant Time Scale,” in **Proc. IEEE INFOCOM’01**, Anchorage, AK, Apr. 2001.
49. H.S. Kim and N.B. Shroff, 2001, “Loss Probability Calculations and Asymptotic Analysis for Finite Buffer Multiplexers,” **IEEE/ACM Trans. Networking**, vol. 9, no. 6, pp. 755–768, Dec. 2001.
50. J.P. Singh, N. Bambos, B. Srinivasan, and D. Clawin, 2002, “Wireless LAN Performance under Varied Stress Conditions in Vehicular Traffic Scenarios,” in **Proc. IEEE Vehicular Technology Conference**, 2002.
51. L. Subramanian and R.H. Katz, 2000, “An Architecture for Building Self-Configurable Systems,” in **Proc. ACM MobiHoc’00**, Boston, USA, Aug. 2000.
52. H. Takagi and L. Kleinrock, 1984, “Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals,” **IEEE Trans. Commun.**, vol. COM-32, no. 3, pp. 246–257, Mar. 1984.
53. C. Yu, K.G. Shin, and B. Lee, 2004, “Power-Stepped Protocol: Enhancing Spatial Utilization in a Clustered Mobile Ad Hoc Network,” **IEEE J. Select. Areas Commun.**, vol. 22, no. 7, pp. 1322–1334, Sept. 2004.
54. Y. Barowski, S. Biaz, and P. Agrawal, 2005, “Towards the Performance Analysis of IEEE 802.11 in Multi-hop Ad-Hoc Networks,” in **Proc. IEEE WCNC’05**, Mar. 2005, pp. 100–106.

APPENDIX A

FIRST APPENDIX TITLE

Put appropriate topic here

This is where you put hardware circuit diagrams, detailed experimental data in tables or source codes, etc..

This appendix describes two static allocation methods for fGn (or fBm) traffic. Here, λ and C are respectively the traffic arrival rate and the service rate per dimensionless time step. Their unit are converted to a physical time unit by multiplying the step size Δ . For a fBm self-similar traffic source, Norros [47] provides its EB as

$$C = \lambda + (\kappa(H)\sqrt{-2 \ln \epsilon})^{1/H} a^{1/(2H)} x^{-(1-H)/H} \lambda^{1/(2H)} \quad (\text{A.1})$$

where $\kappa(H) = H^H(1 - H)^{(1-H)}$. Simplicity in the calculation is the attractive feature of (A.1).

The MVA technique developed in [49] so far provides the most accurate estimation of the loss probability compared to previous bandwidth allocation techniques according to simulation results. Consider a discrete-time queueing system with constant service rate C and input process λ_n with $\mathbb{E}\{\lambda_n\} = \lambda$ and $\text{Var}\{\lambda_n\} = \sigma^2$. Define $X_n \equiv \sum_{k=1}^n \lambda_k - Cn$. The loss probability due to the MVA approach is given by

$$\varepsilon \approx \alpha e^{-m_x/2} \quad (\text{A.2})$$

where

$$m_x = \min_{n \geq 0} \frac{((C - \lambda)n + B)^2}{\text{Var}\{X_n\}} = \frac{((C - \lambda)n^* + B)^2}{\text{Var}\{X_{n^*}\}} \quad (\text{A.3})$$

and

$$\alpha = \frac{1}{\lambda\sqrt{2\pi\sigma^2}} \exp\left(\frac{(C - \lambda)^2}{2\sigma^2}\right) \int_C^\infty (r - C) \exp\left(\frac{(r - \lambda)^2}{2\sigma^2}\right) dr \quad (\text{A.4})$$

For a given ε , we numerically solve for C that satisfies (A.2). Any search algorithm can be used to do the task. Here, the bisection method is used.

Next, we show how $\text{Var}\{X_n\}$ can be determined. Let $C_\lambda(l)$ be the autocovariance function of λ_n . The MVA technique basically approximates the input process λ_n with a Gaussian process, which allows $\text{Var}\{X_n\}$ to be represented by the autocovariance function. In particular, the variance of X_n can be expressed in terms of $C_\lambda(l)$ as

$$\text{Var}\{X_n\} = nC_\lambda(0) + 2 \sum_{l=1}^{n-1} (n-l)C_\lambda(l) \quad (\text{A.5})$$

Therefore, $C_\lambda(l)$ must be known in the MVA technique, either by assuming specific traffic models or by off-line analysis in case of traces. In most practical situations, $C_\lambda(l)$ will not be known in advance, and an on-line measurement algorithm developed in [48] is required to jointly determine both n^* and m_x . For fGn traffic, $\text{Var}\{X_n\}$ is equal to $\sigma^2 n^{2H}$, where $\sigma^2 = \text{Var}\{\lambda_n\}$, and we can find the n^* that minimizes (A.3) directly. Although λ can be easily measured, it is not the case for σ^2 and H . Consequently, the MVA technique suffers from the need of prior knowledge traffic parameters.

APPENDIX B

SECOND APPENDIX TITLE

Put appropriate topic here

Next, we show how $\text{Var}\{X_n\}$ can be determined. Let $C_\lambda(l)$ be the autocovariance function of λ_n . The MVA technique basically approximates the input process λ_n with a Gaussian process, which allows $\text{Var}\{X_n\}$ to be represented by the autocovariance function. In particular, the variance of X_n can be expressed in terms of $C_\lambda(l)$ as

$$\text{Var}\{X_n\} = nC_\lambda(0) + 2 \sum_{l=1}^{n-1} (n-l)C_\lambda(l) \quad (\text{B.1})$$

Add more topic as you need

Therefore, $C_\lambda(l)$ must be known in the MVA technique, either by assuming specific traffic models or by off-line analysis in case of traces. In most practical situations, $C_\lambda(l)$ will not be known in advance, and an on-line measurement algorithm developed in [48] is required to jointly determine both n^* and m_x . For fGn traffic, $\text{Var}\{X_n\}$ is equal to $\sigma^2 n^{2H}$, where $\sigma^2 = \text{Var}\{\lambda_n\}$, and we can find the n^* that minimizes (A.3) directly. Although λ can be easily measured, it is not the case for σ^2 and H . Consequently, the MVA technique suffers from the need of prior knowledge traffic parameters.